Identification of Ornamental Plant which Possess Medicinal Function Based on Its Leaf Shape, Texture, and Color Features

March 2015

Department of Science and Advanced Technology,
Graduate School of Science and Engineering,
Saga University

Indra Nugraha Abdullah

Identification of Ornamental Plant which Possess Medicinal Function Based on Its Leaf Shape, Texture, and Color Features

A doctoral thesis submitted to the Department of Science and Advanced Technology, Graduate School of Science and Engineering, Saga University as a partial fulfilment of the requirements for the Doctoral degree in Department of Science and Advanced Technology.

by

Indra Nugraha Abdullah

Supervisor: Prof. Hiroshi Okumura

Nationality: Indonesia

Previous degrees: Bachelor of Computer Science
Bogor Agricultural University (IPB)
Bogor, Indonesia

Master of Science
Saga University
Saga, Japan

Department of Science and Advanced Technology
Graduate School of Science and Engineering
Saga University
Saga, Japan

THESIS EXAMINATION COMMITTEE

Professor Hiroshi Okumura (Chairman)
Department of Science and Advanced Technology
Graduate School of Science and Engineering
Saga University, Japan

Professor Kohei Arai
Department of Science and Advanced Technology
Graduate School of Science and Engineering
Saga University, Japan

Professor Shin-ichi Tadaki
Department of Science and Advanced Technology
Graduate School of Science and Engineering
Saga University, Japan

Professor Teruya Minamoto
Department of Science and Advanced Technology
Graduate School of Science and Engineering
Saga University, Japan

Associate Professor Koichi Nakayama
Department of Science and Advanced Technology
Graduate School of Science and Engineering
Saga University, Japan

Abstract

Ornamental plant has a primary function as the ornament of one space. Beside that function, it also has another function as a medicine. The medicinal function of this plant mostly lies on its leaf. However, not many people have the knowledge about that function from the ornamental plant. In Indonesia is very common for elder people to use the ornamental leaf to cure the disease, and the number of diseases that can be treated by this leaf is also very broad. The reason we choose the ornamental leaf as a research object because of this leaf has the double functionality not only as ornamental leaf but also as medicinal leaf. The ornamental leaf composed of different shapes, textures, and colors. The Dyadic Wavelet Transform and The Dual-tree Wavelet Complex Wavelet Transform (DT-CWT) are capable of recognizing the differences in the translation aspect and we decide to use it as the shape as well as texture extractor methods. DT-CWT also offers flexibility in the directional selection of its decomposed image. In order to make more competitive comparison, this research involves Zernike Complex Moments as shape feature extractor method with its advantage in rotation changes detection. Although in this research the color is not so significantly different, but there is a different degree of green in our leaf images. Because of that reason plus lighting-invariant property, we apply the grid-approach of HSV Color Histogram for color extractor method. It is found that the Dyadic wavelet transform outperformed as shape and texture extractor method, and together with HSV-color histogram were resulting approximately 90% of correct classification rate.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Human has a duty to preserve the nature. One of the examples is to preserve the existence of the plant. There is a necessity cycle between human and plant. Plant produces oxygen from the photosynthesis process, and human provides carbon dioxide that vital for plant. Logically, human will experience problems when number of the plant is gradually reducing. This research uses leaf from the ornamental plant as a plant that need to be preserved. Ornamental leaf because of its main function as an ornament certainly has sale value. Maintaining the preservation of this plant will give many benefits in many aspects of the human itself.

Based on International Union for Conservation of Nature and Natural Resources, the number of identified plant species in the world which consist of Mosses (M), Ferns and Allies (FA), Gymnosperms (G), Flowering Plants (FP), Red and Green Algae (RGA) is about 307.674 species [1]. Figure 1.1 shows the percentage detail between identified and unidentified species.

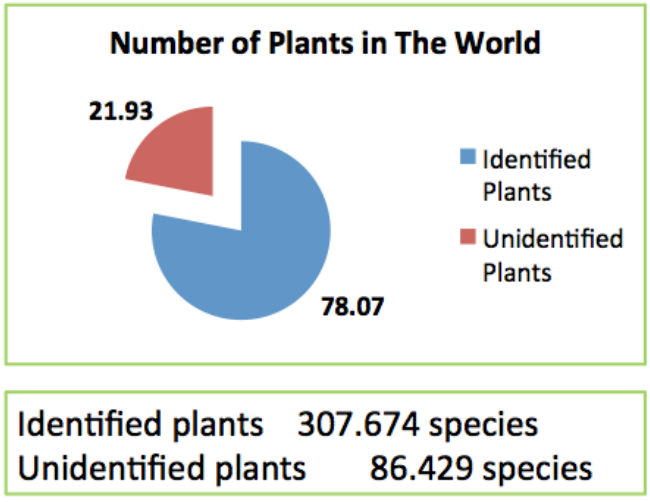


Fig. 1.1: Number of plants in the world

In the table 1.1 presented a huge gap between the flowering plant with the mosses, as the second populated species with the number 268000 and 16236 species, respectively. While the proportion of the mosses, ferns allies and red green algae are not very different. The number of these three categories are approximately about 12000, 10386, and 1052 species.

Table 1.1: Number of identified plant species in the world

Plant Species	Number of species
FP	268000
M	16236
FA	12000
RGA	10386
G	1052

On the other side, the approximate number of unidentified species is 86.429 species as shown in Figure 1.2. It consists of Flowering Plants with 83.400 species, Ferns and Allies with 3.000 species, Mosses with 29 species [2]. Considering the highest number possessed by Flowering Plants, identification of the plants, which also include ornamental plant, has become a challenge for us.

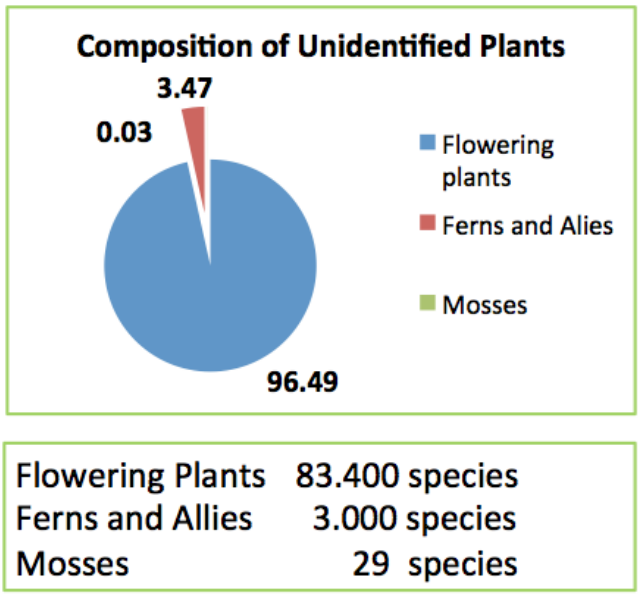


Fig. 1.2: Number of unidentified plants in the worlds

As recognition step of unidentified species, this research is focusing on ornamental leaf that functioned as medicinal leaf. However, only a few people know about its function as a treatment of the disease. In Indonesia, this plant is also easy to find because mostly cultivates in front of the house. If its medicinal function and that easiness are taken into consideration, this plant should be an initial treatment or an option towards full chemical-based medicines. As is known, the long-term use of this kind of medicines is not good for the human body, especially for the liver.

Identification of the leaf image is possibly done through identification of some leaf features, i.e. color, shape, and texture. Previously, most of the researchers were using shape and texture feature to identify the leaf. In 2000, Zhiyong Wang et.a [22] proposed leaf image retrieval using shape features. Leaf shape characterized by centroid-contour distance curve and object eccentricity functions. The eccentricity is used for rank the leaves, and its best-ranked result is further re-rank again using the centroid-contour distance curve. Followed by Park, Hwang, Nam [12] in 2007, they were employing curvature scale scope corner detection method to detect the leaf venation. Categorization of the selected points is managed by calculating the density of feature points. Then, Xiao-feng Wang et.al. [20] conducted a research work to classify leaf image with a complicated background. Segmentation process of the leaf utilizing automatic marker for Watershed method, and the classification process was done by combining 7 Hu moments and 16 Zernike moments.

Du, Zhai, and Wang [5] proposed a research to recognize the plant leaf image based on fractal dimension feature in 2012. They calculated the two-dimensional fractal of the leaf image as well as multiple vein images with one additional ring projection wavelet fractal feature for leaf shape. The most current work was proposed by Wang, Liang, and Guo [21] in 2014, they recognized the plant leaf using dual-scale decomposition and local binary descriptors. The term of dual-scale decomposition relates to the decomposition using lifting wavelet scheme and filtering process using variable-scale Gaussian filters. Local binary descriptors were calculated from the decomposition results and fuzzy k-nearest neighbors as the classifier.

Shape is a substantial part to describe image content, and shape of this ornamental leaf is very diverse. From texture feature, we can obtain one of the important information from leaf called leaf venation. From the aforementioned researchers, they were mainly utilizing the leaf venation as an important feature of a leaf. Through texture feature extraction, we believe that leaf venation can be extracted very well. Subsequently, color information is still considered as an additional feature, despite the dominant color of ornamental leaf is green. In HSV model, we can separate the color information from its intensities. It is become important because of the lighting effect that often interfere the identification process.

This doctoral thesis is organized as follows. Explanation about ornamental plant with its profile that used in this research is in Chapter 2. We present the literature review to assist recollection information regarding the methods in Chapter 3. In Chapter 4, we explain the research methods that consist of shape, texture and color features extraction, also covers the flowchart of this research. Chapter 5 describes the experiment and result that has coverage for the image preprocessing results, invariant checking until classification results. In chapter 6, we show a simple implementation of smart phone application using Android platform. Finally, conclusion will be the last Chapter of this thesis.

Chapter 2

Ornamental Plant

This research emphasis on identification of ornamental plant using its leaf. In Indonesia, this kind of plant is usually cultivated in front of the house. Ornamental leaf on this research is not general ornamental leaf. This ornamental leaf has two general functionalities. First as an ornament in open space, and second as herbal medicine that used to cure some diseases. Image dataset of ornamental leaf on this research is obtained by direct acquisition using a digital camera.

This dataset contains 8 classes with 15 images for each class. The classes are Bay (*syzygium polyanthum*), Cananga (*canagium odoratum, lamk*), Mangkokan (*nothopanax scutellarium merr.*), Jasmine (*jasminum sambac [soland]*), Cocor bebek (*kalanchoe pinnuta*), Vinca (*catharanthus roseus*), Kestuba (*euphorbia pulcherrima, willd*), Gardenia (*gardenia augusta, merr*). These plants are not affected by the seasons, and it will be able to grow-up for one whole year. In order to avoid expensiveness of computation, size of the image is 256 x 256 pixels. Figure 2.1 contains sample images from all classes.

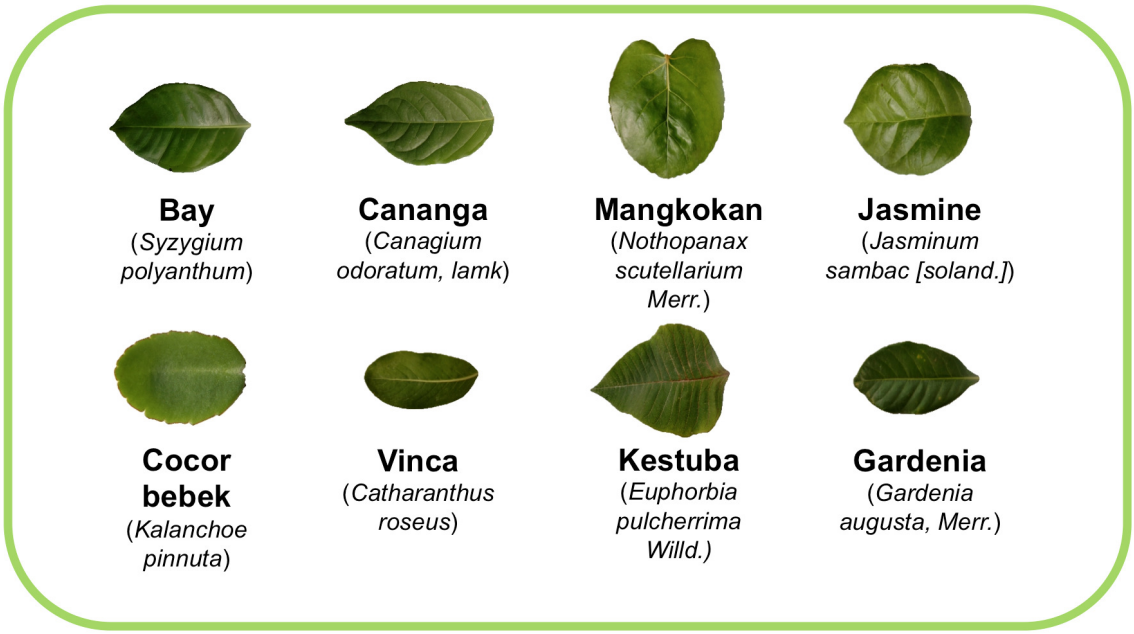


Fig. 2.1: Ornamental leaves used in this research

2.1 Bay

Synonym	Familia
Eugenia polyantha, Wight; E. lucidula, Miq	*Myrtaceae*

Bay grows in the forest and the plateau, but it also cultivates in front of the house. Bay can be found until 1800 m above sea level. Its height can be reached until 25 m, and its stem is around 0.5-1 m. Its leaf's length and width are about 5-15 cm and 3-8 cm, respectively. Corresponding diseases that can be cured by this plant are diarrhea, ulcer, and diabetes. The example of serving way to cure the diarrhea is boiled the water with 15 gr of leaf for about 15 minutes, add a small portion of salt, filter the liquid from its concentrate and apply as drinks.

Fig. 2.2: Bay leaf image

2.2 Cananga

Synonym	Familia
Hook and Thorms	*Annonaceae*

Cananga is included in big-stem plant and can live more than ten years. The tallest one can be reached until 5-20 m. It has a very beautiful flower. Around the flower can be found six leaves with yellow crown and three green leaves. This plant can be easily found in 25-1000 m above sea level. This plant is very useful to cure malaria, asthma, and bronchitis. Pour the hot water together with three florets of a flower, and close the container tightly. After it becomes cold, then separate the water form its concentrate.

Fig. 2.3: Cananga leaf image

2.3 Mangkokan

Synonym	Familia
N. cochleatum, (Lamk), Miq. Polyscias scutellaria, (Burm.F.), Fosb. Panax cochleatum, Dc.	*Araliaceae*

This plant is usually cultivated as an ornamental plant. It can live in a place that has an elevation between 1-200 m above sea level. Its height is around one until three m. The term of mangkokan is come from its shape that similar with mangkok, Indonesia language of bowl. The young leaves are edible as a compliment of the meal. This plant has an efficiency as a relieve from the inflammatory condition inside breast called mastitis, can accelerate the flow

of breast milk and effectively reduce body odor. The dominant serving way is combining this leaf with a small portion of turmeric and cooking oil, apply directly to the inflammation region.

Fig. 2.4: Mangkokan leaf image

2.4 Jasmine

Synonym	Familia
-	*Oleranceae*

Jasmine possesses high number of benefits. It can grow in one whole year and can live very well in a place with elevation from 600 until 800 meter above sea level. It can easily propagate by cuttings. The function of this plant can reduce the flow of breast milk, relieve the fever and cure the sore eyes. The example for serving way is taking a number of leaves together with ten flowers, then squeeze with the hand and apply as a compress to reduce fever.

Fig. 2.5: Jasmine leaf image

2.5 Cocor Bebek

Synonym	Familia
-	*Kalanchoe*

Cocor bebek has an average height about 20-40 cm. Its leaf's length and width are 5-14 cm and 4-8 cm, respectively. The fruit is square-shaped with 4 separated region, and the color is green. The characteristics of seed as follows, round, smooth on the surface and black color. The medicinal function that contained in this plant can cure the burns and sore skin. Pounding about 15 gr of leaf and then apply directly to the skin.

Fig. 2.6: Cocor bebek leaf image

2.6 Vinca

Synonym	Familia
Lochnera rosea, Vinca rosea, Linn. Ammoallis rosea, Small	*Apocynaceae*

Vinca can be separated into two types, the white flower, and red flower types. Maximum height of this plant is approximately about 100 cm. Its leaf is egg-shaped, green color and classified as a single leaf plant. The flower looks like trumpet with the smooth surface. The diseases that can be treated with this plant are very broad, start from hypertension, diabetes, leukemia, asthma, and fever. The serving way for this kind of plant is combining the seven leaves with the flower, boil in the water and apply as drinks before sleep.

Fig. 2.7: Vinca leaf image

2.7 Kestuba

Kestuba is very effective to cure swollen body part, assist the abnormality of menstruation and to stop from bleeding. The way to serve is pounding a number to leaf and apply it to the swollen body part.

Fig. 2.8: Kestuba leaf image

2.8 Gardenia

Synonym	**Familia**
Gardenia Jasminoides, Ellis	*Rubiaceae*

Gardenia is usually cultivated as an ornamental plant because of its aromatic flower. It can grow either in a cold place or even a hot place. However, the most suitable is in a cold place or a location with the elevation more than 400 meter above sea level. Its stem can reach until 1-2 meter. Its leaf has the following characteristics, oval-shape, relatively thick, and a slippery surface. This plant is very helpful to cure sprue, diabetes, and constipation. To treat the constipation, we can boil 2 cups of water with three pieces seed and wait until it is cold, and then apply as drinks.

Fig. 2.9: Gardenia leaf image

Because the number of the diseases which can be treated by using this ornamental plant is very broad. This plant has to be an option towards the fully-chemical based medicine that already known the negative effect for the body for long-term use. The treatment to cultivate this plant is not difficult similarly with another regular plant, no special treatment is needed.

Table 2.1: Summary of medicinal function that contained in ornamental plant

Plant name	**Medicinal Function**
Bay	Diarrhea, ulcer, diabetes
Cananga	Malaria, asthma, bronchitis
Mangkokan	Mastitis, accelerate breast milk, reduce body odor
Jasmine	Fever, sore eyes, reduce breast milk
Cocor Bebek	Burns, sore skin
Vinca	Hypertension, diabetes, leukemia, asthma, fever
Kestuba	Swollen body part, abnormality of menstruation, bleeding
Gardenia	Sprue, diabetes, constipation

Chapter 3

Literature Review

3.1 Dyadic wavelet transformation

Through a reference of translated and dilated wavelets, the continous wavelet transform decomposes one dimensional signals $f \in L^2(\mathbb{R})$

$$\psi_{u,s}(t) = \frac{1}{\sqrt{s}}\psi\left(\frac{t-u}{s}\right). \tag{3.1.1}$$

Translation-invariant wavelet references are composed by sampling the above scale parameter s along an exponential sequence $\left\{v^j\right\}_{j\in\mathbb{Z}}$, during the time keep all translation parameter u. The $v = 2$ is selected to achieve easiness in computer implementation:

$$D = \left\{\psi_{u,2^j}(t) = \frac{1}{\sqrt{2^j}}\psi\left(\frac{t-u}{2^j}\right)\right\}_{u\in\mathbb{R},\, j\in\mathbb{Z}}. \tag{3.1.2}$$

The dyadic wavelet transform of $f \in L^2(\mathbb{R})$ is determined by

$$Wf\left(u, 2^j\right) = \langle f, \psi_{u,2^j}\rangle = \int_{-\infty}^{\infty} f(t)\,\frac{1}{\sqrt{2^j}}\psi\left(\frac{t-u}{2^j}\right)dt = f * \bar{\psi}_{2^j}(u), \tag{3.1.3}$$

with

$$\bar{\psi}_{2^j}(t) = \psi_{2^j}(-t) = \frac{1}{2^j}\psi\left(\frac{-t}{2^j}\right). \tag{3.1.4}$$

This translation-invariant dyadic wavelet transforms has been used widely, especially for pattern recognition and denoising applications. This is possible to apply translation-invariant

references to the multiscale wavelet generators $\phi_n(t) = 2^{-j/2}\psi_{2^j}(t)$. As $\hat{\phi}_n(\omega) = \hat{\psi}\left(2^j\omega\right)$, the Fourier condition means that there is exist two constants $A > 0$ and $B > 0$ such that

$$\forall\omega \in \mathbb{R} - \{0\}, A \leq \sum_{j=-\infty}^{+\infty} \left|\hat{\psi}\left(2^j\omega\right)\right|^2 \leq B, \tag{3.1.5}$$

and as $\Phi f(u,n) = 2^{-j/2}Wf(u,n)$, the below equation shows frame inequality

$$A\|f\|^2 \leq \sum_{j=-\infty}^{+\infty} \frac{1}{2^j}\|Wf\left(u,2^j\right)\|^2 \leq B\|f\|^2. \tag{3.1.6}$$

The dyadic wavelet transform determines a complete and stable representation if the frequency axis is entirely covered by the dilated dyadic wavelets.

Furthermore, if $\tilde{\psi}$ satisfies

$$\forall\omega \in \mathbb{R} - \{0\}, \sum_{j=-\infty}^{+\infty} \hat{\psi} * \left(2^j\omega\right)\hat{\tilde{\psi}}\left(2^j\omega\right) = 1, \tag{3.1.7}$$

the applies to $\tilde{\phi}_n(t) = 2^{-j}\tilde{\psi}\left(2^{-j}t\right)$ proves that

$$f(t) = \sum_{j=-\infty}^{+\infty} \frac{1}{2^j} Wf\left(.,2^j\right) * \tilde{\psi}_{2^j}(t) \tag{3.1.8}$$

3.1.1 Dyadic wavelet design

When the wavelet is properly designed a discrete dyadic wavelet transform can be measured with a fast filter bank algorithm. The reconstruction of these dyadic wavelets is similar to the decomposition of bi-orthogonal wavelet bases. Let h and g be pair of finite-impulse-response filters. The h filter is low-pass filter with a transfer function that complies $\hat{h} = \sqrt{2}$. Similar with the orthogonal and bi-orthogonal wavelet bases, the scaling function is constructed from Fourier transform:

$$\hat{\phi}(\omega) = \prod_{p=1}^{+\infty} \frac{\hat{h}(2^{-p}\omega)}{\sqrt{2}} = \frac{1}{\sqrt{2}}\hat{h}\left(\frac{\omega}{2}\right)\hat{\phi}\left(\frac{\omega}{2}\right). \tag{3.1.9}$$

Let's assume that the above Fourier transform is a finite-energy function so that $\phi \in L^2(\mathbb{R})$, and the correlate wavelet ψ has a Fourier transform which determined by

$$\hat{\psi}(\omega) = \frac{1}{\sqrt{2}}\hat{g}\left(\frac{\omega}{2}\right)\hat{\psi}\left(\frac{\omega}{2}\right). \tag{3.1.10}$$

Both ϕ and ψ have a compact support because of h and g have a finite number of non-zero coefficients. Since the $\hat{\phi}(0) = 1$, the number of vanishing moments of ψ is identical with the number of zeros of $\hat{\psi}(\omega)$ at $\omega = 0$, and identical also with the number of zeros of $\hat{g}(\omega)$ at $\omega = 0$.

3.1.2 Spline dyadic wavelet filters design

The spline dyadic wavelets are often chosen because have the smoothness and symmetrical properties. We can construct the spline dyadic wavelet filters with higher vanishing moments through

$$\hat{h}(\omega) = \sqrt{2}\frac{\hat{\phi}(2\omega)}{\hat{\phi}(\omega)} = \sqrt{2}\left(cos\frac{\omega}{2}\right)^{m+1}\exp\left(\frac{-i\varepsilon\omega}{2}\right). \tag{3.1.11}$$

where m is non-negative integer with

$$\varepsilon = \begin{cases} 1, & \text{if } m \text{ is even} \\ 0, & \text{if } m \text{ is odd} \end{cases}$$

After $\hat{\tilde{h}}$ is selected, the Fourier transform of dual scaling filter $\tilde{h}[k]$ is selected by

$$\hat{\tilde{h}}(\omega) = \hat{h}(\omega) \tag{3.1.12}$$

if we select

$$\hat{g}(\omega) = (-i)^{\tau}\sqrt{2}\left(\sin\frac{\omega}{2}\right)^{r}\exp\left(-i\frac{2-\tau}{2}\omega\right) \tag{3.1.13}$$

where $r = 1, 2$ with

$$\tau = \begin{cases} 1, & \text{if } r{=}1 \\ 0, & \text{if } r{=}2 \end{cases}$$

resulting

$$\hat{g}(\omega) = (-i)^{\tau}\ \sqrt{2}\exp\left(-i\frac{2-\tau}{2}\omega\right)\left(\sin\frac{\omega}{2}\right)^{2-r}\left(\sum_{l=0}^{m}\cos\frac{\omega}{2}\right)^{2-l} \tag{3.1.14}$$

All above described spline dyadic wavelet filters are finite-impulse-response filters. Below table shows the values of the filter when $r = 2$ and $m = 1$, and the notation o is indicating the initial filter.

Table 3.1: Spline dyadic wavelet filters when $r = 2$ and $m = 1$ (2 vanishing moments)

k	$\frac{h^o[k]}{\sqrt{2}}$	$\frac{g^o[k]}{\sqrt{2}}$	$\frac{\tilde{h}^o[k]}{\sqrt{2}}$	$\frac{\tilde{g}^o[k]}{\sqrt{2}}$
-1	0.25		0.25	
0	0.50	-0.25	0.50	0.25
1	0.25	0.50	0.25	1.50
2		-0.25		0.25

As $\hat{g}^o(\omega)$ fulfil

$$\left.\frac{d^k\hat{g}^o(\omega)}{d\omega^k}\right|_{\omega=0} = 0, \quad 0 \le k \le 1,$$
$$\left.\frac{d^2\hat{g}^o(\omega)}{d\omega^2}\right|_{\omega=0} = \frac{\sqrt{2}}{2},$$

the filter $g^o(k)$ has two vanishing moments. Subsequently, we can obtain a highpass analysis filters more than two vanishing moments, as long as $\hat{s}(\omega)$ fulfil

$$\left.\frac{d^k\hat{s}(\omega)}{d\omega^k}\right|_{\omega=0} = 0, \quad 0 \le k \le 1,$$
$$\left.\frac{d^2\hat{s}(\omega)}{d\omega^2}\right|_{\omega=0} = \frac{1}{\sqrt{2}}, \left.\frac{d^2\hat{g}^o(\omega)}{d\omega^2}\right|_{\omega=0} = \frac{1}{2},$$

The $\hat{s}(\omega) = e^{-i\omega}\sin^2\omega/4$ is selected because of above condition and the symmetry of $\hat{s}(\omega)$. Then, the highpass analysis filters with four vanishing moments are obtained.
as $\hat{g}(\omega)$ fulfil

$$\left.\frac{d^k \hat{g}(\omega)}{d\omega^k}\right|_{\omega=0} = 0, \quad 0 \le k \le 3,$$
$$\left.\frac{d^4 \hat{g}(\omega)}{d\omega^4}\right|_{\omega=0} = 3\sqrt{2}$$

the filter $g[k]$ has four vanishing moments.

Table 3.2: Spline filters with 4 vanishing moments

k	$\frac{h[k]}{\sqrt{2}}$	$\frac{g[k]}{\sqrt{2}}$	$\frac{\tilde{h}[k]}{\sqrt{2}}$	$\frac{\tilde{g}[k]}{\sqrt{2}}$
-3			-0.015625	
-2		0.015625	-0.093750	
-1	0.25	0.031250	0.265625	
0	0.50	-0.265625	0.687500	0.25
1	0.25	0.437500	0.265625	1.50
2		-0.265625	-0.093750	0.25
3		0.031250	-0.015625	
4		0.015625		

Let's design another filters via setting the value of $r = 1$ and $m = 2$.
as $\hat{g}(\omega)$ fulfil

$$\hat{g}(0) = 0, \quad \left.\frac{d\hat{g}(\omega)}{d\omega}\right|_{\omega=0} = -\frac{\sqrt{2}}{2}i,$$

the filter $g(k)$ has one vanishing moment. We can expand to the higher vanishing moments of highpass analysis filters. If we select $\hat{s}(\omega) = -i \sin \omega/2$, from this condition

$$\hat{s}(0) = 0, \quad \left.\frac{d\hat{s}(\omega)}{d\omega}\right|_{\omega=0} = -\frac{1}{\sqrt{2}}, \quad \left.\frac{d\hat{g}(\omega)}{d\omega}\right|_{\omega=0} = -\frac{-i}{2}$$

then, following table is obtained.

Table 3.3: Spline dyadic wavelet filters when $r = 1$ and $m = 2$ (1 vanishing moment)

k	$\frac{h[k]}{\sqrt{2}}$	$\frac{g[k]}{\sqrt{2}}$	$\frac{\tilde{h}[k]}{\sqrt{2}}$	$\frac{\tilde{g}[k]}{\sqrt{2}}$
-2				-0.03125
-1	0.125		0.125	-0.21875
0	0.375	-0.5	0.375	-0.68750
1	0.375	0.5	0.375	0.68750
2	0.125		-0.125	0.21875
3				0.03125

Table 3.4: Spline filters with 3 vanishing moments

k	$\frac{h[k]}{\sqrt{2}}$	$\frac{g[k]}{\sqrt{2}}$	$\frac{\tilde{h}[k]}{\sqrt{2}}$	$\frac{\tilde{g}[k]}{\sqrt{2}}$
-3			0.0078125	
-2		-0.03125	0.546875	-0.03125
-1	0.125	-0.09375	0.2890625	-0.21875
0	0.375	-0.56250	0.1484375	-0.68750
1	0.375	0.56250	0.1484375	0.68750
2	0.125	0.09375	0.2890625	0.21875
3		0.03125	0.546875	0.03125
4			0.0078125	

As $\hat{g}(\omega)$ fulfil

$$\left.\frac{d^k \hat{g}(\omega)}{d\omega^k}\right|_{\omega=0} = 0, \quad 0 \le k \le 2,$$
$$\left.\frac{d^3 \hat{g}(\omega)}{d\omega^3}\right|_{\omega=0} = \frac{3\sqrt{2}}{2} i,$$

the filter $g[k]$ that shown in above table has three vanishing moments.

3.2 Dual-tree complex wavelet transform

The N-dimensional Dual Tree Complex Wavelet Transform (DT-CWT) is a very effective approach for the N-dimensional wavelet-based signal processing like analyzing and processing oriented singularities in image. It keeps the advantages from 1-dimensional dual tree CWT and achieves additional property.

The separable 2-dimensional wavelet transform less efficient for edge, but efficient enough to represent the line and curve singularities in the image. One of the motives of the DT-CWT is development of 2-dimensional multiscale transform that can represent the line and curves in the image more efficient. Curvelets and steerable pyramids transforms are two examples of transformation that able to isolate edges with different orientation in the different subbands.

The separable 2-dimensional wavelet transform decomposes image into three different subbands. LH subband is vertical direction wavelet, HL is horizontal direction wavelet, and HH is diagonal direction wavelet. The following equation shows the detail

$$\begin{aligned}
\psi_1(m,n) &= \phi(m)\psi(n) \quad \textit{LH detail} \\
\psi_2(m,n) &= \psi(m)\phi(n) \quad \textit{HL detail} \\
\psi_3(m,n) &= \psi(m)\psi(n) \quad \textit{HH detail}
\end{aligned} \tag{3.2.1}$$

where m is denoted first dimension data or row, n is second dimension data or column. The ϕ is process of convolution using low-pass filter and the ψ using the high-pass filter. For example, the LH detail is obtained from the convolution using low-pass filter along row, and using high-

pass filter along column. This wavelet transform has a lack of directional selectivity because of it only support $+45^0$, -45^0 and mix from both.

The reason this problem appeared is its real function spectrum are the two-sided spectrum both for the vertical detail or the horizontal detail. It is not avoidable looking at the fact that the spectrum will contain passbands in all four corners of the two-dimensional plane. In order to distinguish between the vertical and horizontal details become difficult whether in the frequency domain or spatial domain.

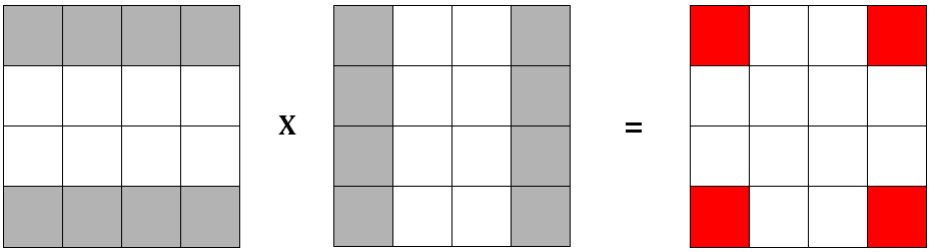


Fig. 3.1: Lack of directional selectivity from 2-dimensional separable wavelet

From this point, the details of DT-CWT will be explained. The DT-CWT is resulting oriented wavelets through the below equation

$$\begin{aligned}\psi(m,n) &= [\psi_l(m) + j\psi_l(m)]\,[\psi_h(n) + j\psi_h(n)] \\ &= \psi_l(m)\,\psi_l(n) - \psi_h(m)\,\psi_h(n) + j\,[\psi_l(m)\,\psi_h(n) + \psi_l(m)\,\psi_h(n)]\end{aligned} \tag{3.2.2}$$

where $\psi(m)$ is a complex and the wavelet defined by $\psi(m) = \psi_l(m) + j\psi_h(m)$. The $\psi_l(m)$ and $\psi_h(n)$ are the HH details using low-pass filters, high-pass filters, respectively. The implementation to construct subband detail is similar with separable wavelet which described earlier. The complex 2-dimensional wavelet is oriented because the analytic 1-dimensional wavelet's spectrum is located in only one region of its frequency plane. If we consider the real part of then we can expand into more than one region

$$Re\{\psi(m,n)\} = \psi_l(m)\,\psi_l(n) - \psi_h(m)\,\psi_h(n) \tag{3.2.3}$$

where *Re* denotes real part. The spectrum of this wavelet with direction -45^0 is now located in two regions of the 2-dimensional frequency plane because of the spectrum of real function has to be symmetric with its origin.

For direction $+45^0$ lets consider the 2-dimensional wavelet $\psi_2(m,n) = \psi(m) * \psi(n)$ and the * denotes complex conjugate. Similar with the previously described -45^0, the equations become

$$\begin{aligned}\psi(m,n) &= [\psi_l(m) + j\psi_l(m)] * [\psi_h(n) + j\psi_h(n)] \\ \psi(m,n) &= [\psi_l(m) + j\psi_l(m)][\psi_h(n) - j\psi_h(n)] \\ &= \psi_l(m)\psi_l(n) + \psi_h(m)\psi_h(n) + j[\psi_l(m)\psi_h(n) - \psi_l(m)\psi_h(n)]\end{aligned} \tag{3.2.4}$$

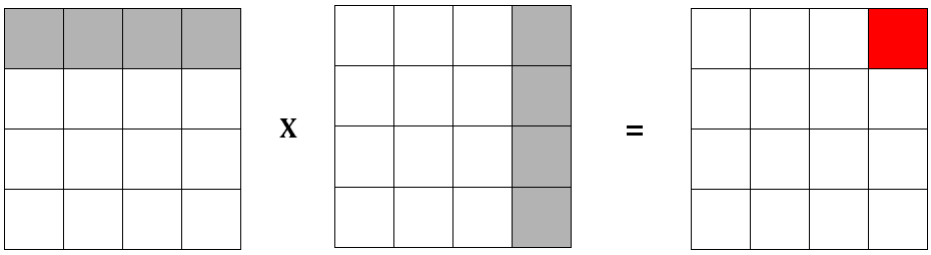


Fig. 3.2: Support of fourier transform for this CWT in minus 45 degree

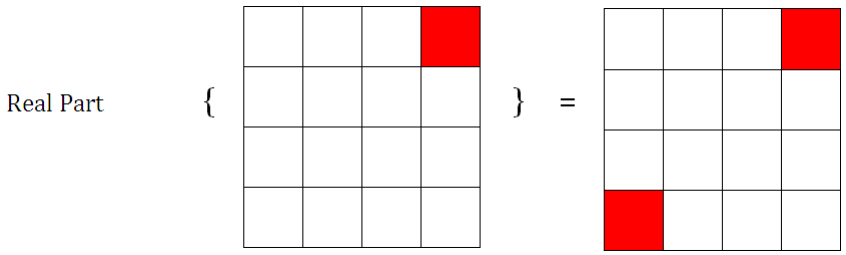


Fig. 3.3: Minus 45 degree spectrum of real wavelet in CWT

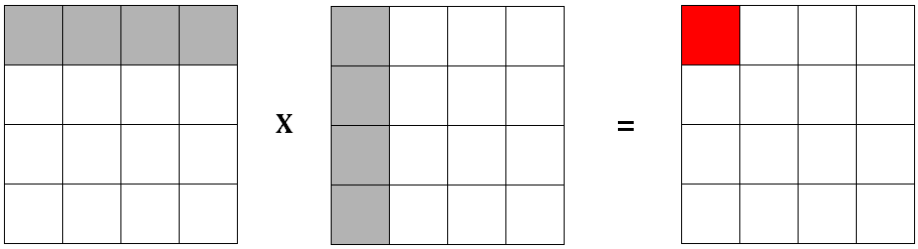


Fig. 3.4: Support of fourier transform for this CWT in 45 degree

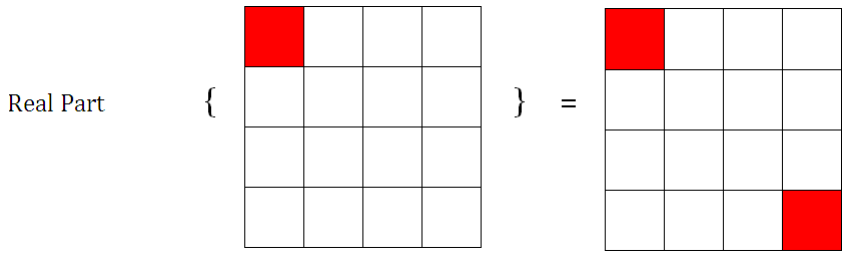


Fig. 3.5: 45 degree spectrum of real wavelet in CWT

It can be repeated similarly with -45^0 and 45^0 orientations to obtain another four orientations of oriented 2-dimensional wavelets which are $\phi(m)\psi(n)$, $\psi(m)\phi(n)$, $\phi(m) * \psi(n)$ and $\psi(m) * \phi(n)$. Finally, the following six wavelets are achieved

$$\begin{aligned} \psi_i(m,n) &= \frac{1}{\sqrt{2}}\left(\psi_{1,i}(m,n) - \psi_{2,i}(m,n)\right), \\ \psi_{i+3}(m,n) &= \frac{1}{\sqrt{2}}\left(\psi_{1,i}(m,n) + \psi_{2,i}(m,n)\right) \end{aligned} \tag{3.2.5}$$

for $i = 1, 2, 3$ and the following are its separable 2-dimensional wavelet bases

$$\begin{aligned} \psi_{1,1}(m,n) &= \phi_l(m)\psi_l(n), & \psi_{2,1}(m,n) &= \phi_h(m)\psi_h(n), \\ \psi_{1,2}(m,n) &= \psi_l(m)\phi_l(n), & \psi_{2,2}(m,n) &= \psi_h(m)\phi_h(n), \\ \psi_{1,3}(m,n) &= \psi_l(m)\psi_l(n), & \psi_{2,3}(m,n) &= \psi_h(m)\psi_h(n) \end{aligned} \tag{3.2.6}$$

From this point will describe a 2-dimensional transform which oriented and complex. That advantages are obtained because of the oriented complex 2-dimensional wavelet transform is four-times expansive. Beside the mentioned advantages, this wavelet also has full shift-invariant property. The way to achieve spectrum support of the imaginary part in the 2-dimensional frequency is similar with the previously described real part and the minus 45 degree oriented wavelet is defined as follow

$$Im\{\psi(m,n)\} = \psi_h(m)\psi_l(n) - \psi_l(m)\psi_h(n) \tag{3.2.7}$$

The first and second terms are the HH wavelet of separable 2-dimensional wavelet using high-pass filter then low-pass filter, the HH wavelet using low-pass filter then high-pass filter, respectively. This applies also to the other six directions which are $\psi(m) * \psi(n)$, $\phi(m)\psi(n)$, $\psi(m)\phi(n)$, $\phi(m) * \psi(n)$ and $\psi(m) * \phi(n)$, then resulting the following six oriented wavelets

$$\begin{aligned} \psi_i(m,n) &= \frac{1}{\sqrt{2}}\left(\psi_{3,i}(m,n) + \psi_{4,i}(m,n)\right), \\ \psi_{i+3}(m,n) &= \frac{1}{\sqrt{2}}\left(\psi_{3,i}(m,n) - \psi_{4,i}(m,n)\right) \end{aligned} \tag{3.2.8}$$

for $i = 1, 2, 3$ and the following are its separable 2-dimensional wavelet bases

$$\begin{aligned} \psi_{3,1}(m,n) &= \phi_h(m)\psi_l(n), & \psi_{4,1}(m,n) &= \phi_l(m)\psi_h(n), \\ \psi_{3,2}(m,n) &= \psi_h(m)\phi_l(n), & \psi_{4,2}(m,n) &= \psi_l(m)\phi_h(n), \\ \psi_{3,3}(m,n) &= \psi_h(m)\psi_l(n), & \psi_{4,3}(m,n) &= \psi_l(m)\psi_h(n) \end{aligned} \tag{3.2.9}$$

The magnitude of the complex wavelet is the nearly circular bell-shaped function and oriented at the same angle for real as well as imaginary parts of each complex wavelet.

3.3 Zernike complex moments

Simple construction of rotation invariant is one of the main reasons when applying the moments. Rotation is transformed into shift where a change of phase can be further eliminated by the multiplication of appropriate moments. This idea engages the application of moments orthogonal on disk.

Initially, Zernike polynomials were proposed by Zernike in 1934. In terms of information redundancy and reconstruction capability, his moment formulation becomes to be one of the most popular. In image analysis area, Zernike Moments (ZMs) were introduced by Teague. Teague used the facts that ZMs kept their magnitude under arbitrary rotation and applied it to achieve rotation invariants in an image. He presented that second and third order of ZMs are equal to the Hu invariants moments in terms of geometric moments.

A complete basis set defined on the unit disc $\left(x^2 + y^2\right) \leq 1$ is formed from complex polynomials. By using this complex polynomials, the ZMs can be constructed. Lets define two-dimensional ZMs with A_{mn}

$$A_{mn} = \int_x \int_y f(x,y) \left[V_{mn}(x,y)\right] * dxdy \quad x^2 + y^2 \leq 1 \tag{3.3.1}$$

and $m = 0, 1, 2, ..., \infty$ and n describe the order and the repetition, respectively. The * denotes the complex conjugate while $f(x,y)$ is the described function. Above equation must fulfil following conditions

$$m - |n| = even, \quad |n| \leq m, \tag{3.3.2}$$

and $A*_{mn} = A_{m,-n}$ is true.

The Zernike polynomials $V_{mn}(x,y)$ can be expressed in polar coordinates with

$$V_{m,n}(r,\theta) = R_{m,n}(r) \exp(jn\theta) \tag{3.3.3}$$

where (r,θ) are described in the unit disc, $j = \sqrt{-1}$ and $R_{mn}(r)$ are imaginary unit and the orthogonal radial polynomial, respectively. The radial polynomial defines with

$$R_{mn}(r) = \sum_{s=0}^{\frac{m-|n|}{2}} (-1)^s F(m,n,s,r) \tag{3.3.4}$$

and

$$F(m,n,s,r) = \frac{(m-s)!}{s!\left(\frac{m+|n|}{2} - s\right)!\left(\frac{m-|n|}{2} - s\right)!} \tag{3.3.5}$$

where the radial function will be symmetric is respect to repetition, $R_{mn}(r) = R_{m,-n}(r)$. The following equations are the examples of selected radial polynomials

$$R_{00}(r) = 1 \qquad R_{11}(r) = r$$

$$R_{20}(r) = 2^2 - 1 \qquad R_{22}(r) = r^2$$

$$R_{31}(r) = 3r^3 - 2r \qquad R_{33}(r) = r^3$$

$$R_{40}(r) = 6r^4 - 6r^2 + 1 \qquad R_{42}(r) = 4r^4 - 3r^2 \qquad R_{44}(r) = r^4$$

$$R_{51}(r) = 10r^5 - 12r^3 + 3r \qquad R_{53}(r) = 5r^5 - 4r^3 \qquad R_{55}(r) = r^5$$

ZMs is very useful for pattern recognition in two-dimensional image because of its rotation-invariant ability. From here, will describes the implentation of ZMs in image and its equation becomes

$$A_{mn} = \sum_x \sum_y f(x,y)\left[V_{mn}(x,y)\right] * \quad x^2 + y^2 \leq 1 \tag{3.3.6}$$

The calculation of ZMs can be done through mapping the image into unit disc using polar coordinates with the center of the image is the origin of the unit disc. The pixels outside the unit disc will be ignored. The method to convert cartesian coordinate *x,y* to polar coordinate *r,θ* as follows

$$x = r\cos\theta \quad y = r\sin\theta \tag{3.3.7}$$

where

$$r = \sqrt{x^2 + y^2} \quad \theta = \tan^{-1}\left(\frac{y}{x}\right) \tag{3.3.8}$$

3.4 Color histogram

Histogram is the foundation for many spatial domain processing methods. The histogram of an image with gray-level in the range [0, N-1], where N is the maximum intensity value, is a discrete function $hist\,(g_k) = p_k$, g_k defines the kth gray-level and p_k is the number of pixels which have gray-level g_k in the image. This histogram is very often used for image enhancement regarding the gray-level characteristics: dark, light, low and high contrast.

The following image is presented as the example of histogram calculation with the discrete function defined by:

$$[0, 16] = [0, 5] \cup [6, 12] \cup [13, 16]$$

$$range = bin_1 \cup bin_2 \cup bin_3$$

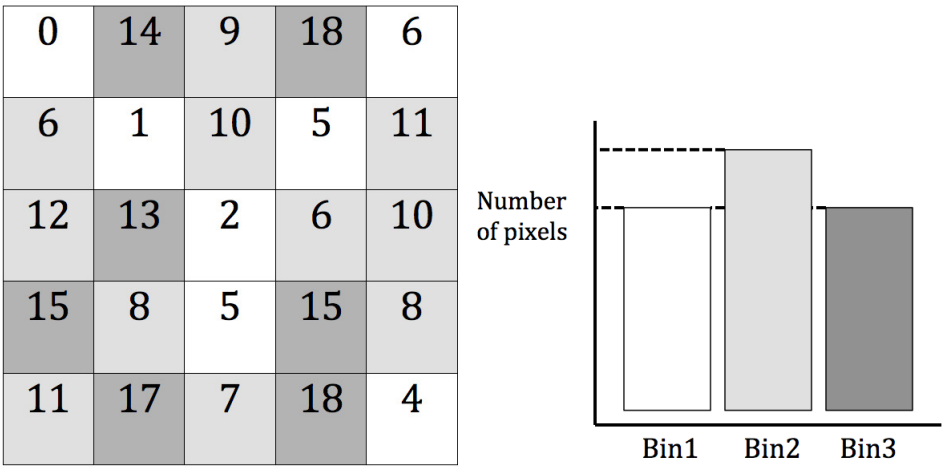


Fig. 3.6: Example of histogram calculation

where the gray-level is 16 level and $bins$ are the number of subdivisions in each data dimension, for the above example $bins = 3$. The process is only counting the discrete function for specified pixels. In example, $hist\,(bin_1) = hist\,(0) + hist\,(1) + hist\,(2) + hist\,(3) + hist\,(4) + hist\,(5) = 7$. The same calculation also applies for bin_2 and bin_3.

The aforementioned approach can also be extended to the color information. Many researchers have been used color histogram when they are dealing with the tasks like image retrieval and image identification. The reason for the usage is because it possesses the simplicity to be calculated and very robust against many color changes in the image. Actual definition of color histogram is vector that the entry is the number of pixels in the image of a specified color. The colors are mapped into a discrete colorspace containing m number colors. In the RGB image, which is the most used colorspace, the discretization process can be done through utilization of the number of most significant bits (MSB) per color channel.

Chapter 4

Method

4.1 Image preprocessing

The basic motivation to do this process is to obtain the proper shape of the leaf. The problems from originally taken leaf image using digital camera are the existence physical abnormality from the leaf itself like of hole/s on leaf surface, improper technique when taking the image, and differences of lighting aspect because of sunlight. The improper technique is often associated with the taken leaf image's size exceeding the border of the image and overlapping leaves condition in the image.

Most of the above problems can be solved using the sequential strategy as presented in figure 4.1. This strategy consists of changing the color image to gray-level image, then applies the blur and binarization methods, followed by the morphological operations, and the edge drawing method as the last step. The important points, why we apply the blur and binarization methods, are to cover the hole/s on the surface and to omit the relatively strong lighting aspect found in the image, respectively. The morphological operations have task to create perfect leaf shape and the edge drawing method to overcome the disconnected leaf contour problem.

Another problem is overlapping leaves in one image. If we continue to use this condition of an image will not resulting a proper result, because it will affect its shape, texture, and color features extraction. This research put an aim to separate the overlapping leaves image utilizing Watershed algorithm. The complete steps of this separation process are described with the following steps:

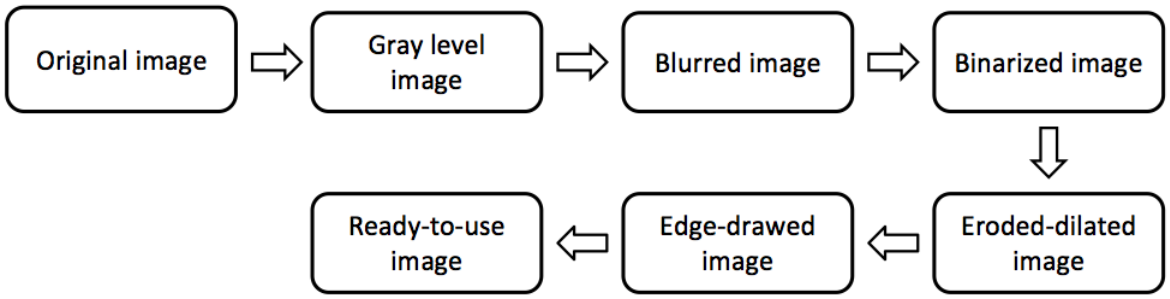


Fig. 4.1: Image preprocessing of the leaf image

1. Create structuring element (SE) for morphological operation process
2. Set the anchor point in the middle of SE
3. Erode the image until two separated shapes is obtained
4. Set the markers, foreground marker for biggest shape, background marker for background plus smaller shape
5. Do a watershed transform with previously decided markers
6. Select the biggest shape marker that associated with foremost single leaf

all parameters such as the size of structuring element and the number of iteration for morphological operation are decided carefully to achieve a perfect single leaf.

Through aforementioned strategy, we are expecting the leaf image that used in this research the proper leaf image and were able to achieve a good identification system. The term of proper is related to a complete shape with texture detected image.

4.2 Research flow

The feature extraction process can be divided into three subsequent steps. Firstly is shape feature extraction that utilize the ornamental leaf shape as the extraction object. Secondly is texture feature extraction that has an aim to extract the main information that contained in the leaf called leaf venation. Thirdly is color feature extraction that still important regardless the color distribution in our dataset is limited to green only.

For shape and texture features extraction will be fully supported by the wavelet transformation. Wavelet transform delivers the advantages to the leaf shape, so be able to discriminate any translation changes that happened in the image. Using the additional step, we can make the shape feature elements to have an ability in scale changes. Since the wavelet can separate the high-frequency and low-frequency components, we can simply choose the required components only. Additionally, it can be extended to multilevel transformation and will also extend the flexibility of the desired level.

Zernike with its rotation invariant property will be a great competitor in shape feature extraction step. The image information will be mapped into the unit disc using Zernike formula with different order and repetition. These differences affect in its advantages which able to detect the rotation changes of the same image or class. In this research, we employ the fast calculation of radial polynomials using recursive method. We are expecting the tight competition between wavelet transform and this method to be the one of selected shape feature extraction method.

Almost all images in the dataset have green color, but still we find that there is a different degree of green between the images. The HSV color space is engaged in this research because of numerous reasons. This color space has ability to divide the image information into two parts, color information (*chroma*) and the intensities or brightness (*luma*). It has been proven that the HSV color space is very powerful in detection of lighting changes and reduce the negative effects of the shadows existence.

As presented in figure 4.2, the initial stage is the image preprocessing for all images inside the datasets. The image needs to be preprocessed because of aforementioned reasons. Followed by alteration from the color image into the gray-level image. From this point, we will enter the feature extraction process. Start from shape feature extraction, detection of the leaf contour from gray-level image as the initial step, then store the detected contour pixels in 1-dimensional data. Shape feature vector as a final product from this shape feature extraction is obtained via application of 1-dimensional wavelet transformation.

For texture feature extraction, we directly do a 2-dimensional wavelet transform to the gray-level image. Afterward, the calculation of statistical-related features element of the transformed image for all details, start from approximation, horizontal, vertical and diagonal. These values are the representation values of texture feature. Slightly different with shape and texture

features extraction. The color feature extraction has an input from the original image. First step is to change the RGB color space into HSV color space. As the global approach histogram is often resulting unsatisfied results, so we decided to use grid-approach histogram. This approach will slice the image into four different parts, and each part will have its corresponding histogram. These histograms are the representation of color feature vector.

Before the normalization process, we append the shape, texture, and color features vectors into one vector. Originally, the values will be ranging in between 0 and 1. Classification as a further stage is completed by using SVM classifier with different kernel functions and parameters. SVM classifier is initially created as a binary classifier, but it can be extended to a multiclass classifier.

In figure 4.3, a minor different point can be found in the shape extraction step. This time, the Zernike complex moments will have a duty as shape feature extractor method. The beginning step is binarize the gray-level image, followed by the computation radial polynomial and basis function. Then those calculated values are being projected to the image and calculation of its moments as a final step. Save the moment's values as shape feature elements.

4.3 Shape and texture feature extraction

4.3.1 Dyadic wavelet transform

In the literature review has described about the Dyadic wavelet design. We can develop to the implementation from that point. The filters of the discrete Fourier transform denote with $h[k]$, $g[k]$, $\tilde{h}[k]$, and $\tilde{g}[k]$ by $\hat{h}(\omega)$, $\hat{g}(\omega)$, $\hat{\tilde{h}}(\omega)$, and $\hat{\tilde{g}}(\omega)$, respectively. Assume that the Fourier transforms fulfil this condition:

$$\hat{\tilde{h}}(\omega)\,\hat{h}*(\omega) + \hat{\tilde{g}}(\omega)\,\hat{g}*(\omega) = 2, \quad \omega \in [-\pi, \pi], \tag{4.3.1}$$

the * represents complex conjugate. Through this condition, the function $f \in L^2(R)$ can be developed into

$$f(t) = \sum_{j=-\infty}^{+\infty} \frac{1}{2^j} Wf\left(., 2^j\right) * \tilde{\psi}_{2^j}(t) \tag{4.3.2}$$

From the condition in 4.3.1, it is obtained

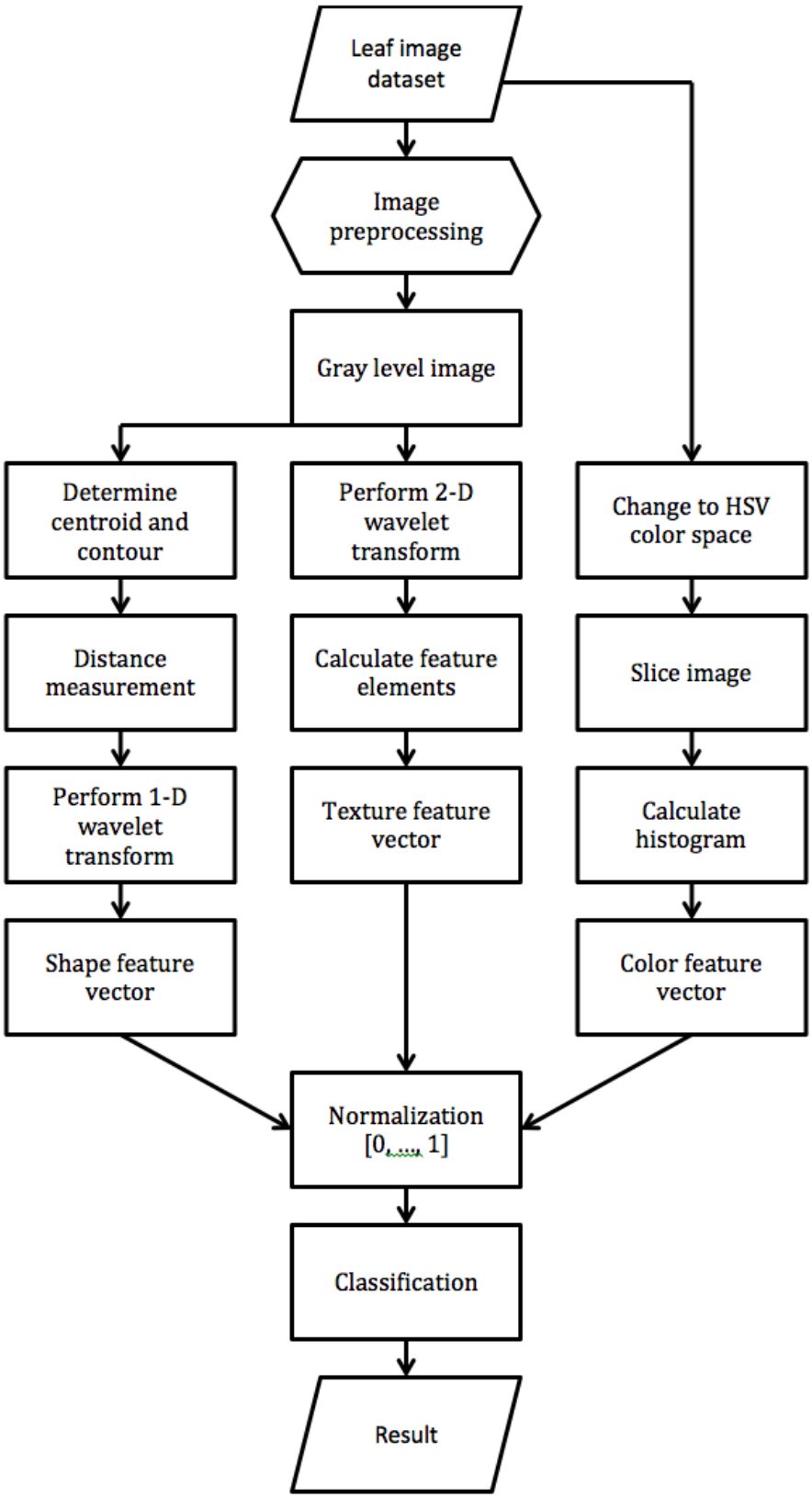


Fig. 4.2: Flowchart of this research (shape feature extraction by Wavelet transform)

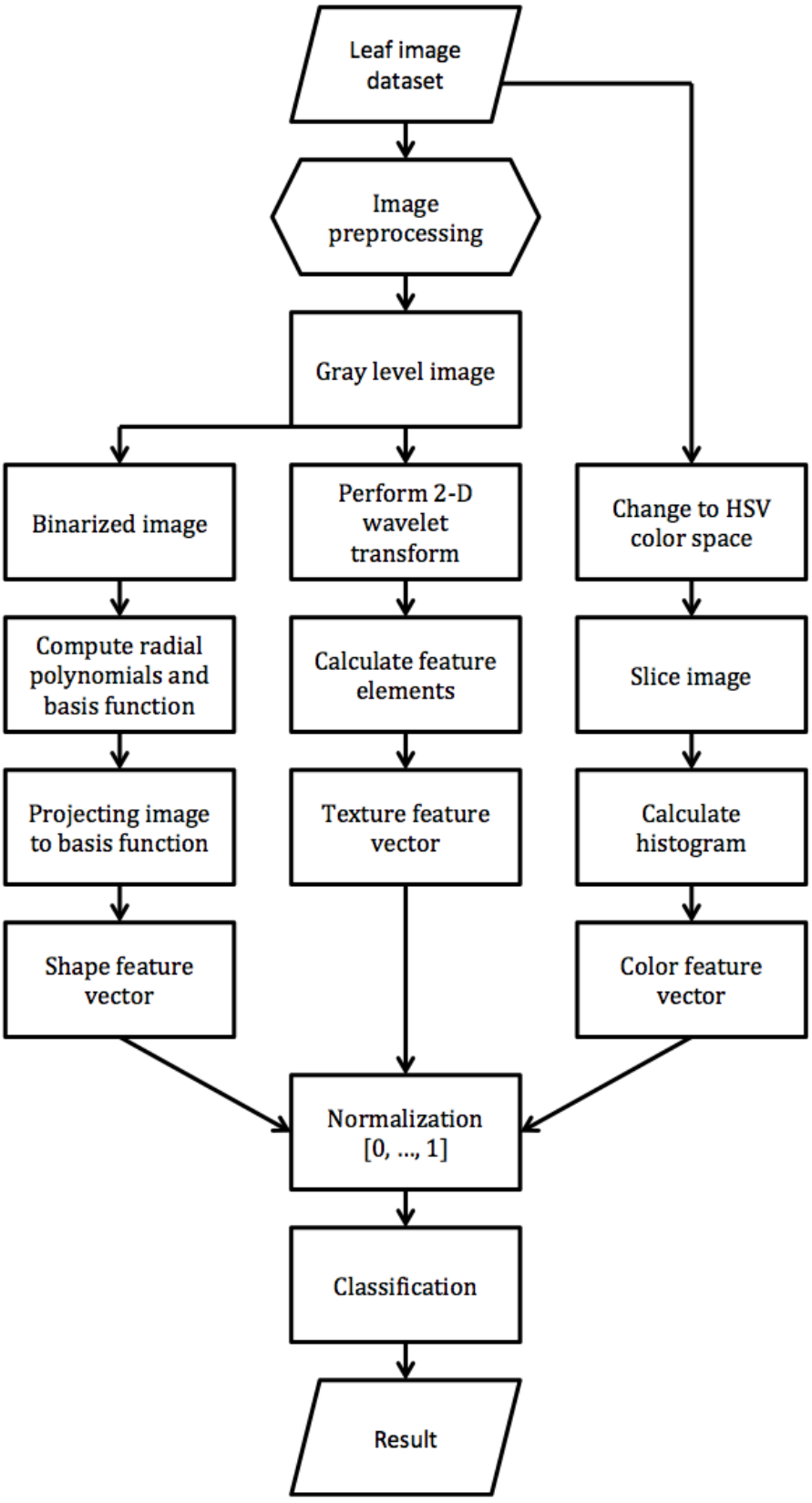


Fig. 4.3: Flowchart of this research (shape feature extraction by Zernike Moments)

$$
\begin{aligned}
a_{j+1}[n] &= \sum_k h[k]\, a_j\left[n + 2^j k\right], \quad j = 0, 1, ..., \\
d_{j+1}[n] &= \sum_k g[k]\, a_j\left[n + 2^j k\right], \quad j = 0, 1, ...,
\end{aligned}
\tag{4.3.3}
$$

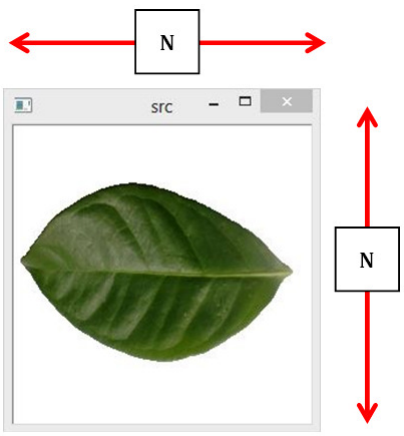
In the case of 2-dimensional data or image, we can apply the equations in 4.3.3, which originally for 1-dimensional data, to the row and then column of an image.

The Dyadic wavelet transform is different with another regular downsampling wavelet such as the Daubechies wavelet or bi-orthogonal wavelet, which often fails in the tasks like feature extraction and image denoising. The Dyadic wavelet is only sampling the scale parameter of continuous wavelet transform and doesn't sample the translation factor, make it possesses the translation-invariant property. This property becomes important when we face the task to extract the most important feature of the leaf called leaf venation. This method can guarantee the arbitrary translation changes in image are not make any effects in the extraction result. The different decomposition results of Dyadic and downscaling wavelet transforms are presented in figure 4.4.

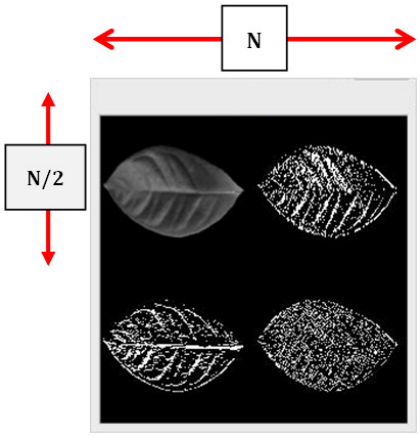
4.3.2 Dual-tree complex wavelet transform

Problems such as lack of shift invariances and poor directional selectivity that usually appears in DWT can be solved effectively using the DT-CWT. Besides that ability, DT-CWT also has limited redundancy and efficient order-N computation. By doubling the sampling rate at each level of the tree, we can obtain approximate shift invariance with real DWT. This doubling process is done by eliminating the downsampling by two after level 1, and this is equal to two parallel fully-decimated trees. The filters in one tree must supply half a sample different of delays from the other tree. Odd-length in one tree and even-length in the other are required for linear phase. The image below is the dual-tree filters for the CWT.

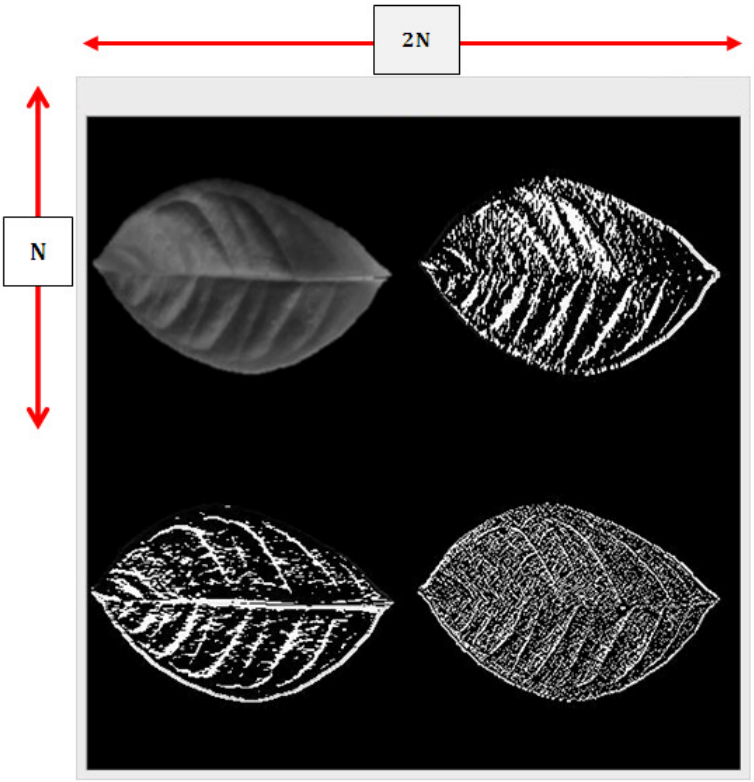
From figure 4.5, tree A is real DWT that gives the real part of wavelet transform, and tree B is real DWT that gives the imaginary part. These real DWTs use different sets of filters. h_0, h_1 and g_0, g_1 denote low-pass or high-pass filter pair for upper filter bank and low-pass or high-pass filter bank for lower filter bank, respectively.



(a) Original leaf image



(b) Downsampling wavelet decomposed image



(c) Dyadic wavelet decomposed image

Fig. 4.4: Difference between the downsampling and Dyadic decomposed images

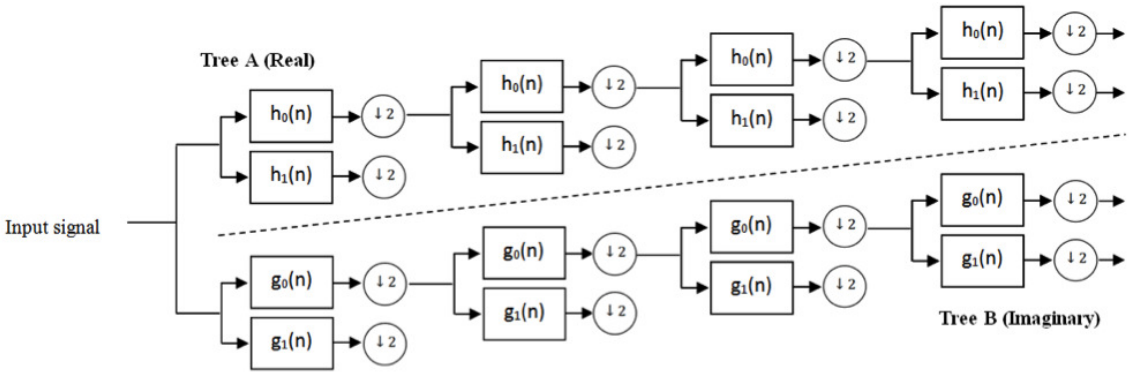


Fig. 4.5: DT-CWT in details

Different with the real DWT, which only have three sub-images in total. DT-CWT decomposes three sub-images for each spectral quadrant 1 and 2 and will have six bandpass sub-images of the complex coefficient at each level. Because of the complex wavelet filters can separate positive from negative frequency vertically and horizontally, the orientation for these six sub-images will cover $\pm 75^0$, $\pm 45^0$ and $\pm 15^0$.

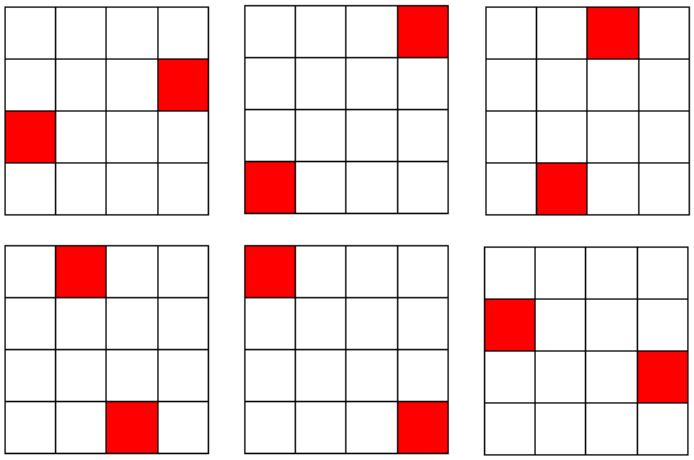


Fig. 4.6: Directional selectivity provided by DT-CWT

In figure 4.6, the advantages of DT-CWT, which called directional selectivity, is applicable to the real part, as well as the imaginary part, and the feature can be extracted easily from those parts. Additionally, this DT-CWT also possesses nearly rotation invariant property, illustrated by the red points in the image. The approximate shift invariant and this near rotation invariant become our important points when deciding to use DT-CWT as feature extractor method.

4.3.3 Shape feature extraction

Wavelet-derived extraction

For contour detection, we apply the work that originally proposed by Suzuki and Abe in 1985. This method has been proven effectively for binary image. They used the 4-(8-) neighborhood in the 4-(8-) connected case. Let denotes an input image with $I = \{i_{xy}\}$ and set the sequential number of the current border (NC) to 1. Reset the sequential number of the newly found border or parent (NN) to 1 every time the scanning process to a new row is begin. The following steps are the details of how this algorithm works:

1. First condition has to be selected:

   - If $i_{x,y} = 1$ and$i_{x,y-1} = 0$ then the pixel (x, y) is the border of an outer border, increment the NC; $(x, y-1) \rightarrow (x_2, y_2)$
   - Else if $i_{x,y} =\geq 1$ and$i_{x,y+1} = 0$, then the pixel (x, y) is the border of an hole border, increment NC; $(x, y+1) \rightarrow (x_2, y_2)$
   - Otherwise; proceed to step 4

2. Decide the parent of the current border based on the types of the newly found border with its NN (i.e. last border meets the current row).

3. Trace the detected border from the starting point (x, y) via this sub steps:

   (a) From the current point, (x_2, y_2), find a nonzero pixel in the neighbourhood of (x, y) in clockwise manner. Set the firstly found nonzero pixel with (x_1, y_1). Assign -NN to $i_{x,y} = 1$, if no nonzero pixel is found and proceed to step 4.

   (b) Set $(x_1, y_1) \rightarrow (x_2, y_2)$ and $(x, y) \rightarrow (x_3, y_3)$

   (c) From the next element of (x_2, y_2), find a nonzero pixel in the neighbourhood of (x_3, y_3) in counter clockwise manner. Set the firstly found nonzero pixel with (x_4, y_4).

   (d) Alter the value of i_{x_3,y_3} of the pixel (x_3, y_3) as follows:

   - If the $i_{x_3,y_3+1} = 0$ then $-NC \rightarrow i_{x_3,y_3+1}$
   - If the $i_{x_3,y_3+1} \neq 0$ then $i_{x_2,y_2} = 1$ and $-NC \rightarrow i_{x_2,y_2}$

- Otherwise; do not change the i_{x_3,y_3}

(e) If $(x_4, y_4) = (x, y)$ and $(x_3, y_3) = (x_1, y_1)$ then proceed to step 4. Otherwise, $(x_3, y_3) \rightarrow (x_2, y_2)$, $(x_4, y_4) \rightarrow (x_3, y_3)$ and return to sub step (c)

4. $|i_{xy}| \rightarrow NN$ if $i_{xy} \neq 1$ and continue the scanning process from the pixel $(x, y + 1)$.

The above algorithm will be stopped when the scanning process reaches the lower right corner of the input image.

The detected contour is further saved as a 1-dimensional data. The detected pixels will be traced in the clockwise manner and save its pixels as the shape feature elements. In order to gain the efficiency, the number of the detected contour pixel can be minimized in the image preprocessing stage. For example, at that stage we can employ the image thinning to achieve the maximum representation of a leaf shape. Let denotes $I = \{(i_x, i_y)\}$ as an input image, the centroid or central moments is $C = \{(c_x, c_y)\}$, $E = \{(e_x, e_y)\}$ as a contour vector and the number of the detected contour pixels is n. Thus, the complete elements of the contour vector will be $E = \{(e_{x_1}, e_{y_1}), (e_{x_2}, e_{y_2}), (e_{x_3}, e_{y_3}), ..., (e_{x_n}, e_{y_n})\}$ corresponding with figure 4.7. The number of n is totally depended on the shape of the leaf itself. Further step is distance measurement between the contour pixel to the centroid pixel, and save the results as distance vector $D = \{d_1, d_2, d_3, ..., d_n\}$. This vector will be an input for wavelet transformation. The wavelet result as the final results will be $S = \{s_1, s_2, s_3, ..., s_n\}$. For more details please refer to figure 4.8.

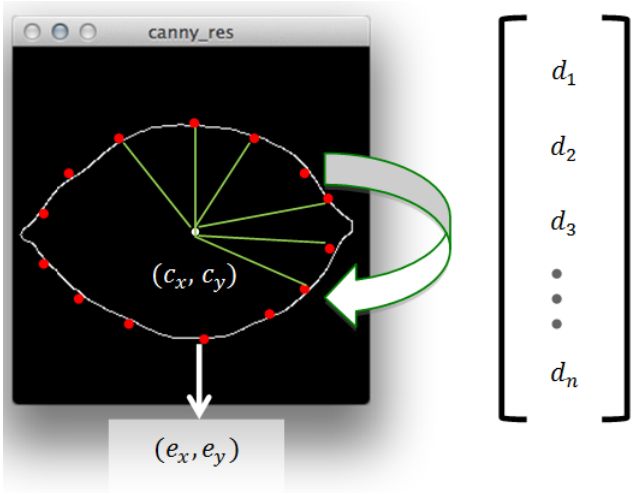


Fig. 4.7: Contour pixels are saved in 1-dimensional matrix

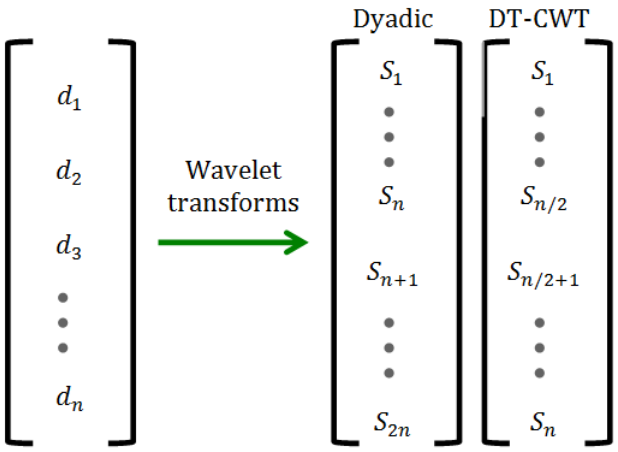


Fig. 4.8: Shape feature vector extracted by the different wavelet transforms

In figure 4.8, the term of shape feature element in this subsection is the wavelet and scaling coefficients. These coefficients are constructed from wavelet transformation process. Because the scope is 1-dimensional of wavelet transform, so we only will have two distinct components, single low-frequency component, and single high-frequency component. The Dyadic wavelet and DT-CWT are resulting different numbers of coefficient because the Dyadic is not a downsampling wavelet transform. Scale invariant is obtained by normalizing all the coefficients with its maximum coefficient. We can briefly assume the accuracy of the shape elements that discard its maximum element will about equal with the complete shape elements. The elements of the shape vector for Dyadic wavelet become:

$$\begin{aligned} S_{dyad_1} &= \{\frac{s_2}{s_1}, \frac{s_3}{s_1}, \frac{s_4}{s_1}, ..., \frac{s_n}{s_1}\}, \\ S_{dyad_2} &= \{\frac{s_{n+2}}{s_{n+1}}, \frac{s_{n+3}}{s_{n+1}}, \frac{s_{n+4}}{s_{n+1}}, ..., \frac{s_{2n}}{s_{n+1}}\} \end{aligned} \tag{4.3.4}$$

for the DT-CWT the elements of shape vector become:

$$\begin{aligned} S_{cwt_1} &= \{\frac{s_2}{s_1}, \frac{s_3}{s_1}, \frac{s_4}{s_1}, ..., \frac{s_{n/2}}{s_1}\}, \\ S_{cwt_2} &= \{\frac{s_{n/2+2}}{s_{n/2+1}}, \frac{s_{n/2+3}}{s_{n/2+1}}, \frac{s_{n/2+4}}{s_{n/2+1}}, ..., \frac{s_n}{s_{n/2+1}}\} \end{aligned} \tag{4.3.5}$$

Moments-derived extraction

We can substitute the integral operation with the summation operation to calculate the zernike complex moments in the image domain. The discrete Zernike complex moments for an image with size $N \times N$ is defined with the following equation:

$$\begin{aligned} A_{m,n} &= \frac{m+1}{\lambda_N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y)\, V^*_{m,n}(x,y) \\ &= \frac{m+1}{\lambda_N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y)\, f(x,y)\, R_{mn}\left(r_{x,y}\right) \exp\left(jn\theta_{x,y}\right) \end{aligned} \tag{4.3.6}$$

where λ_N is normalization factor, $j = \sqrt{-1}$ and $R_{mn}\left(r_{x,y}\right)$ are imaginary unit and the orthogonal radial polynomial, respectively. Transformed distance $r_{x,y}$ and the phase $\theta_{x,y}$ at the pixel position (x, y) are showed with following equations:

$$r_{x,y} = \frac{\sqrt{(2a - N + 1)^2 + (2b - N + 1)}}{N} \tag{4.3.7}$$

$$\theta_{x,y} = tan^{-1}\left(\frac{N - 1 - 2a}{2a - N + 1}\right) \tag{4.3.8}$$

The magnitude of Zernike complex moments from original image with its rotated image will be equal.

This work is only using the low-order of Zernike complex moments because two reasons, which are computation inexpensive and not sensitive to the noise. The table 4.1 shows the proposed low-order Zernike complex moments. However, we still find a problem at the calculation of radial polynomials. As a solution, we prefer to use the recurrence relation of that polynomials. This method only requires two terms, i.e. $R_{m,m-2}, R_{m,m-4}, ..., R_{m,0}$, for the subsequent polynomials of the same order but with different repetition. Only the $R_{m,m}$ and $R_{m,m-2}$ polynomials are needed for the evaluation of power r, don't need the remaining polynomials.

The procedure of radial polynomials' calculation is as below:

1. $R_{mm}(r) = r^m$
2. $R_{m,m-2} = mr^m - (m-1)r^{m-2}$
3. $R_{m,n} = H_1 R_{m,n+4}(r) + (H_2 + \frac{H_3}{r^2}) R_{m,n+2}(r), n = m-4, m-6, ..., 1 \ (or \ 0)$

where

$$\begin{aligned} H_1 &= \frac{(n+4)(n+3)}{2} - (m+4)H_2 + \frac{H_3(m+n+6)(m-n-4)}{8} \\ H_2 &= \frac{H_3(m+n+4)(m-n-2)}{4(n+3)} + (n+2) \\ H_3 &= -\frac{4(n+2)(n+1)}{(n+m+2)(n-m)} \end{aligned} \tag{4.3.9}$$

Table 4.1: Low order of Zernike complex moments

Order (m)	**Iteration (n)**
1	1
2	0, 2
3	1, 3
4	1, 2
5	1, 3, 5
6	1, 2, 4, 6
7	1, 3, 5, 7
8	1, 2, 4, 6, 8
9	1, 3, 5, 7, 9
10	1, 2, 4, 6, 8, 10

4.3.4 Texture feature extraction

Transformation of an image with square size $N \times N$, which is included in the spatial domain, to the frequency domain through wavelet transformation is resulting four distinct details with two different categories. First category is low-resolution or low-frequency component represented by approximation detail. Second is high-resolution or high-frequency components represented by horizontal, vertical and diagonal details. Lets denotes the A_n, H_n, V_n and D_n with n represents n-th decomposition level are the approximation, horizontal, vertical and diagonal details, respectively.

The decomposed image from the original image I or A_0 results the following sub-images:

$$[A_n, \{H_m, V_m, D_m\}_{m=1,\ldots,3}] \tag{4.3.10}$$

where m denotes the high-resolution sub-image in a specified direction at n-th level. The total number of sub-images that used for each decomposition level are 3 high-frequency sub-images and 1 low-frequency sub-image. It will be obtained 4 sub-images for decomposition level 1, 4 sub-images for level 2 and so on. The wavelet energy formula for high-resolution details at the n-th level is follows:

$$\begin{aligned} E_{h,m} &= \sum_{x=0}^{N} \sum_{y=0}^{N} (H_m(x,y))^2 \\ E_{v,m} &= \sum_{x=0}^{N} \sum_{y=0}^{N} (V_m(x,y))^2 \\ E_{d,m} &= \sum_{x=0}^{N} \sum_{y=0}^{N} (D_m(x,y))^2 \end{aligned} \tag{4.3.11}$$

these energies show the strength of the different details in different direction at the n-th level.

Another texture feature elements are the mean and standard deviation. Mean value can be considered as a good estimator for the center of distribution while standard deviation measures how close the whole data to its mean value. Through mean value, the system will easily know which part of the leaf texture of a leaf image is important. Standard deviation has task to support the mean value via finding another best representation of a leaf texture. The mean and standard deviation are calculated from the sub-images (data) in the different direction as well as the level. Thus, to get the correct and systematically values, the coefficient of variation is added as a last element. The strong point possesses by that element can support the scatter of variables stated in distinct units.

$$\begin{aligned} \mu &= \frac{1}{N}\sum_{i=1}^{N} x_i, \\ \sigma &= \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i-\mu)^2}, \\ cv &= \frac{\sigma}{\mu} \end{aligned} \tag{4.3.12}$$

X denotes a finite number of data that consists of $x_1, x_2, x_3, ..., x_N$, while the μ, σ, cv are the mean, standard deviation and coefficient of variation, respectively. Visual representation is delivered with the image below:

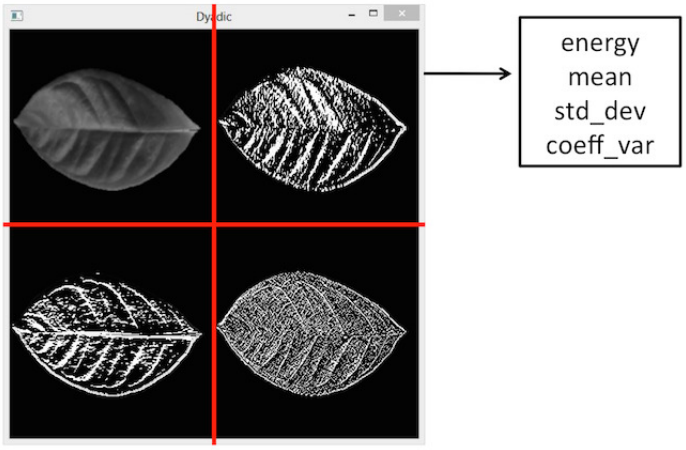


Fig. 4.9: Texture feature elements represented by energy and statistical related values

4.4 Color feature extraction

The HSV color space has its ultimate advantages, which can separate the color information and its intensity information. It means the separation of lighting impact which related to the brightness from the color information also can be easily achieved. The lighting impact on the leaf surface will be dominantly affected by the overall results. The lighting impact on this scope is sunlight that often appeared on the leaf and had minimum impact on the morning as well as afternoon. However, this system should have an ability also in any situation and condition.

Division process from one whole image become four parts is taken because of the test aspects in this research consist of scaling and translation aspects. If the image is translated or scaled to the arbitrary position, then the system will probably identify that the modified image is not belong to the correct class. As a solution and in order to gain a better accuracy but still maintain an efficiency, the original image is sliced into four different regions. Each region will have its corresponding histogram that expected to have the capability in translated and scaled images. As presented in figure 4.10, this process begins with transformation of the original which has RGB color space into HSV color space. Further process is image division that originally one image into four different sub-images.

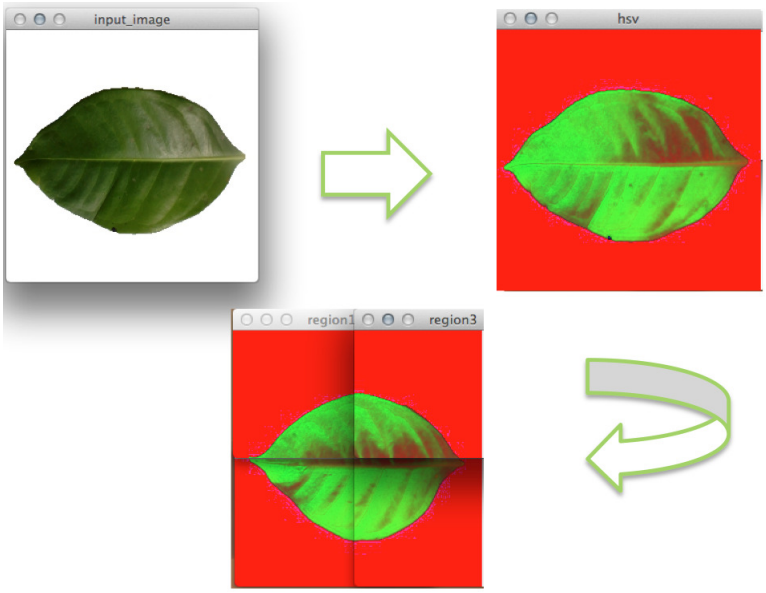


Fig. 4.10: Division of one whole image and color space changing

In the simple words, the color histogram is obtained by counting the number of pixels of specified color. Two crucial steps at the histogram calculation, first is quantization of the color information and second is computation of the histogram itself. At the histogram calculation, we only calculate the color information that are the Hue and Saturation and another unimportant information is discarded. The region numbering is in the clockwise manner. The color information in each pixel will be quantized and put its result in the appropriate color bin. Because the color distribution is limited only to green, we utilize small number of bins.

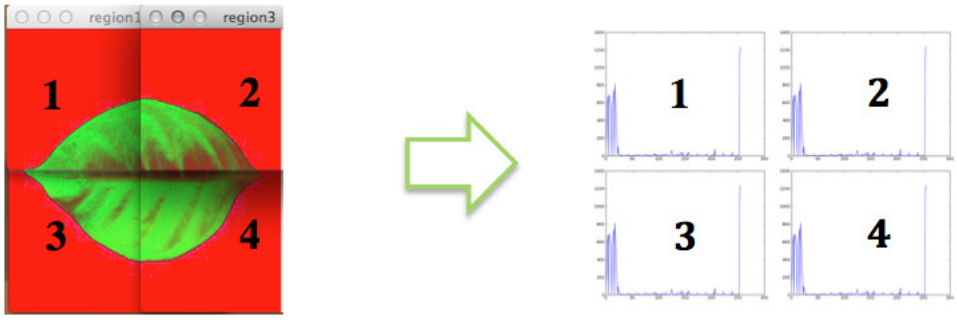


Fig. 4.11: Each region has its own histogram

4.5 Classification

This research engages SVM classifier for classification purpose. SVM has two general advantages, first is the method that designed for a linear classifier able to create non-linear decision boundary. Second is the flexibility that provided by the kernel functions. Some important aspects before we use the SVM are the data preprocessing, type of the kernel function, and the best parameter setting for the SVM as well as the kernel function. SVM tries to find the largest margin between the support vectors as seen in figure 4.12 and figure 4.13.

Usually, the data is separated into the training data and test data. The bigger proportion is always given to the training data to gain better knowledge of its data. Each instance in training data has one-class label with the numbers of attribute. The aim of SVM is to create a model that can predict the class label of test data based on its attribute with maximum margin between class. SVM relates to two types of margin. First, hard-margin to classify the linearly separable data and second, soft-margin for non-linearly separable data. For actual case, the soft-margin cases are often happened instead of the hard-margin. With increasing the dimension of the data in the classification process is the way how the soft-margin works.

Lets label the training data with instance-label pairs (x_i, y_i) where $i = 1, ..., l$ and $x_i \in R^n$, $y \in \{1, -1\}^l$, the SVM needs a solution of the optimization problem in hard-margin case as follows:

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2} w^T w \\ subject\ to \quad & y_i \left(w^T (x_i) + b \right) \geq 1, \ i = 1, ..., l. \end{aligned} \tag{4.5.1}$$

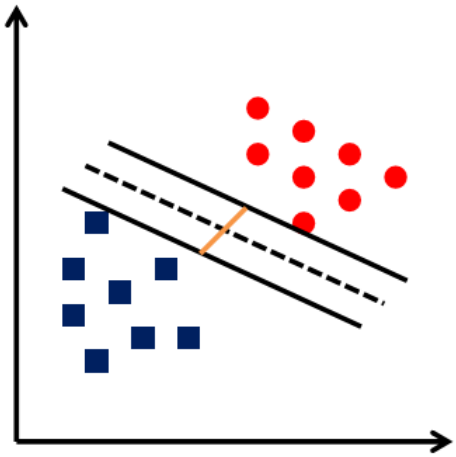


Fig. 4.12: SVM with small margin between class in process to find large margin

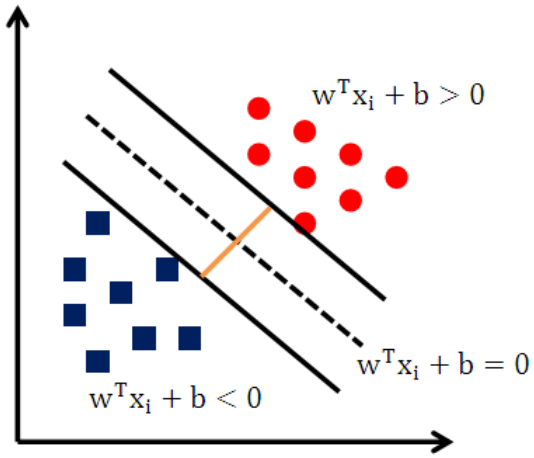


Fig. 4.13: SVM with large margin

where for soft-margin case is:

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi_i \\ subject\ to \quad & y_i \left(w^T \phi(x_i) + b \right) \geq 1 - \xi_i, \xi \geq 0. \end{aligned} \tag{4.5.2}$$

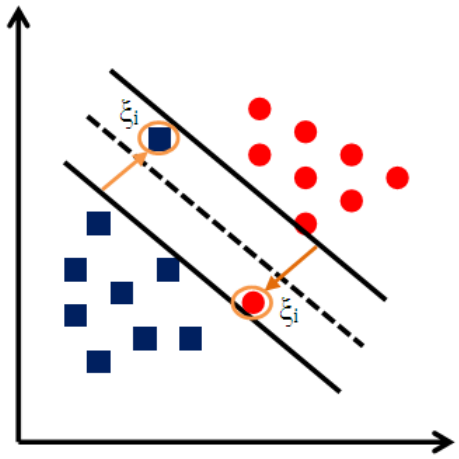


Fig. 4.14: SVM of soft margin case that utilizes margin error

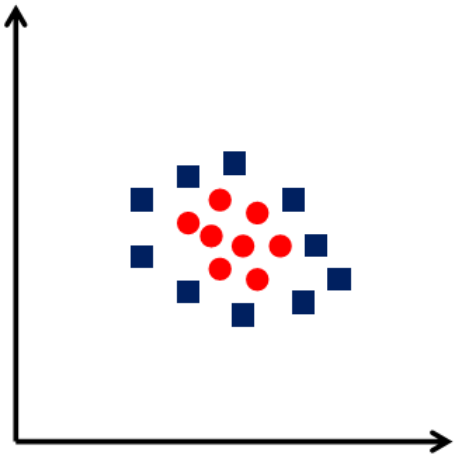


Fig. 4.15: Example of non-linearly separable data that can be handled by soft-margin SVM

In figure 4.14, the SVM employs ξ denotes the margin error or slack variable. It allows the data instance to be in the margin $0 \leq \xi_i \leq 1$ or to be misclassified with $\xi_i > 1$. Figure 4.15 and figure 4.16 are presented as the examples of soft-margin case.

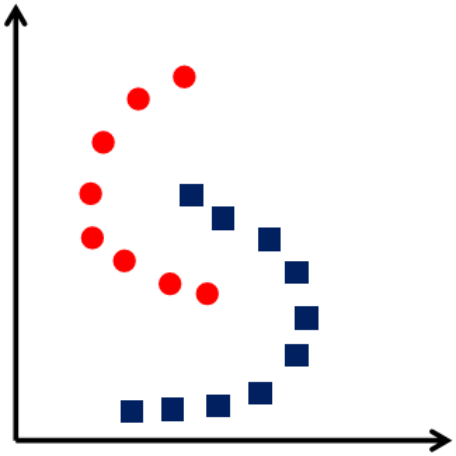


Fig. 4.16: Another examples of non-linearly separable data

By using the function ϕ, the vectors x_i are mapped into a higher dimensional space and searches a hyperplane with the maximal margin in that space. The penalty of an error term denotes with $C > 0$. Many researchers have already introduced some kernel functions, the following functions are some example of the basic kernel functions:

- linear: $K\left(x_i, x_j\right) = x_i^T x_j$
- polynomial: $K\left(x_i, x_j\right) = \left(\gamma x_i^T x_j + r\right)^d, \gamma > 0$
- radial basis function: $K\left(x_i, x_j\right) = \exp\left(-\gamma \|x_i - x_j\|^2\right), \gamma > 0$
- sigmoid: $K\left(x_i, x_j\right) = \tanh\left(\gamma x_i^T x_j + r\right)$

where γ, r, and d are the kernel parametes. The important notes before using the SVM classifier to our data are:

- Do a preprocessing data such as normalization [0,..,1]
- Use RBF kernel function is sometimes more advantageous
- Perform the cross validation to get the optimum parameter settings for C and γ

4.5.1 SVM with one-against-one strategy

The one-against-one (OAO) method is very often also called with pairwise-coupling, all-pairs or round-robin. The OAO constitutes an SVM for each pair of classes, so if the class number is n classes, it will create $n(n-1)/2$ SVMs. These SVMs are trained to discriminate the samples data from one class with the samples from another class. Each SVM votes for one class and a class with the maximum voting will be chosen as the result. In our data, we have eight classes of ornamental plant, and it means the application of this strategy will construct $(8 \times (8-1))/2$ or 28 SVMs. The following image describes the OAO schema for our data that class labels are represented by alphabetical labels.

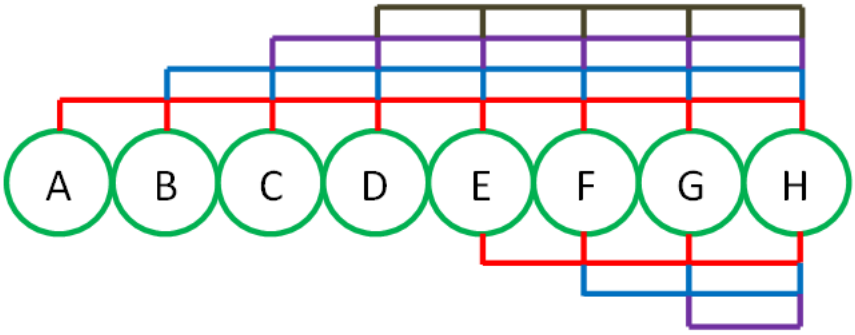


Fig. 4.17: One-againts-one SVM schema

The further task after mapping the output of each SVM into probability is the task to decide the global posterior probabilities, denotes with $\hat{P}(\omega_i|x)$, as functions of the local posterior probabilities, denotes with $\hat{P}\left(\omega_i|f_{i,j}(x)\right)$. The $f_{i,j}(x)$ denotes the output of the SVM. This output is used to determine class ω_i from class ω_j.

Lets consider the probabilities for all classes ω_i

$$\sum_{j=1,j\neq i}^{n} P\left(\omega_{i,j}|x\right) - (n-2)\,P(\omega_i|x) = 1, \tag{4.5.3}$$

where $\omega_{i,j}$ denotes the union of class ω_i and ω_j. Through

$$\hat{P}\left(\omega_i|f_{i,j}(x)\right) \approx \frac{P(\omega_i|x)}{P\left(\omega_{i,j}|x\right)}, \tag{4.5.4}$$

we can derive it to the following equation:

$$\hat{P}(\omega_i|x) = \frac{1}{\sum_{j=1,j\neq i}^{n} \frac{1}{\hat{P}\left(\omega_i|f_{i,j}(x)\right)} - (n-2)}. \tag{4.5.5}$$

each estimate needs to be divided to guarantee the sum of all probabilities is 1, $\hat{P}(\omega_i|x)$ divide by $\sum_{i=1}^{n} \hat{P}(\omega_i|x)$.

4.5.2 Various test aspects

In relation to measuring the performance comprehensively, we involve various test aspects. These aspects covered translation, scaling, rotation and lighting changes aspects. The translation, scaling, rotation to test the performance of the wavelet transformations and Zernike complex moments. Then, lighting changes will be tested by the color extraction method. While the perspective aspect is an aspect that often happens in real life and tested by using the combination of the three features extraction methods. Sample image for each test aspect can be observed in the following image:

(a) Original aspect

(b) Translation aspect

(c) Scaling aspect

(d) Rotation aspect

(e) Lighting aspect

Fig. 4.18: Various involved test aspects

Chapter 5

Experiment and Result

5.1 Image preprocessing

There are several problems that often arise for leaf image identification systems. Firstly, lighting impact on the leaf surface. This problem happened because of ornamental plant located in the outside area where the sun as the main light source will surely give dominant lighting impact to the leaf. Secondly, the physical abnormality of leaf shape such as hole/s on the leaf surface. This abnormality is often caused by the insect or the seasonal changing. Thirdly, false photo acquisition technique when acquire leaf image. Shape of the leaf is not taken completely, some parts of the leaf shape are exceeding the photo borders. Last problem is overlapping leaves condition between the foreground leaf and background leaves. These aforementioned problems must be formerly solved before entering the main process.

To overcome the light impact and holes problem, we can use a strategy that consists of several sub-process. First is color space changing from the color image to gray-level image. Followed by application of blur method to the gray-level image. After we obtained the blurred image, then we make use of binarization method. After these three sub-processes, the system can well-recognize the leaf shape. The below image displays the detail.

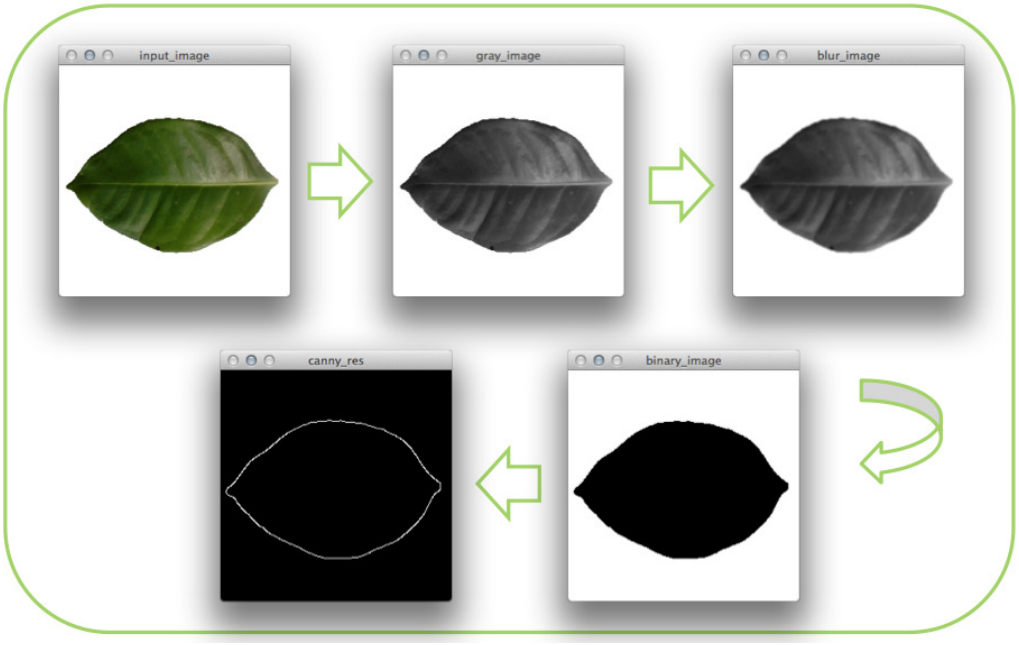


Fig. 5.1: Image preprocessing of the leaf image

The purpose of blurring method is for remove the hole/s on a leaf surface while the binarization method has a purpose to omit strong lighting impact and connect the leaf contour. Please refer to figure 5.2 for the details. Another problem is false methodology of leaf image acquisition. This problem arises from the time leaf shape is not perfectly taken inside the leaf image. Some parts of the leaf are exceeding the photo borders and happened in the left, upper, right and below part of borders. Single sequential strategy is needed to solve this problem. This strategy involves three methods, first is edge drawing method to connect the disconnected leaf contour. Second is morphological erosion operation to make the contour more thick. After the contour is connected, then we apply the image thinning to obtain the optimal representation of the leaf contour. The details is presented in figure 5.3 and the solution result in figure 5.4.

(a) Strong lighting impact and disconnected contour **(b)** Hole/s still detected

Fig. 5.2: The problems can be handled using this pre-processing strategy

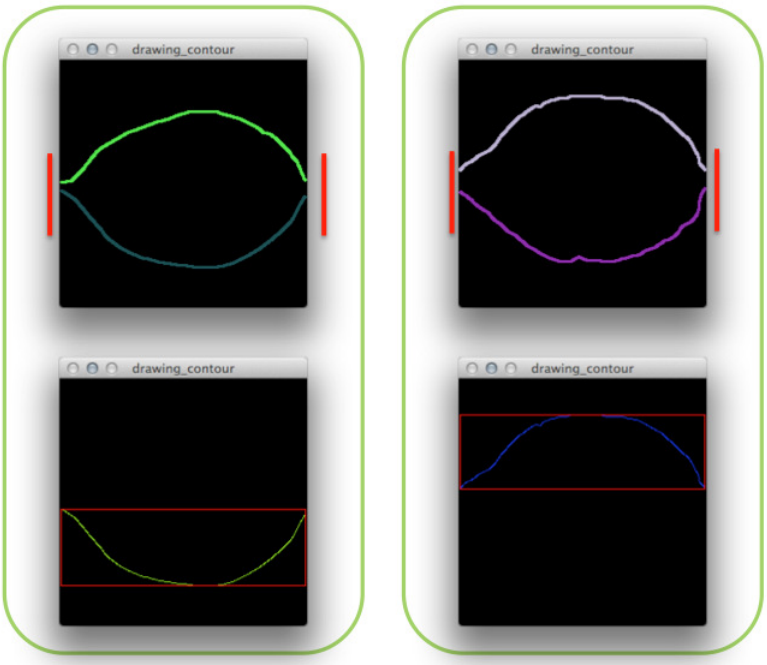


Fig. 5.3: Effect of false photo taking technique

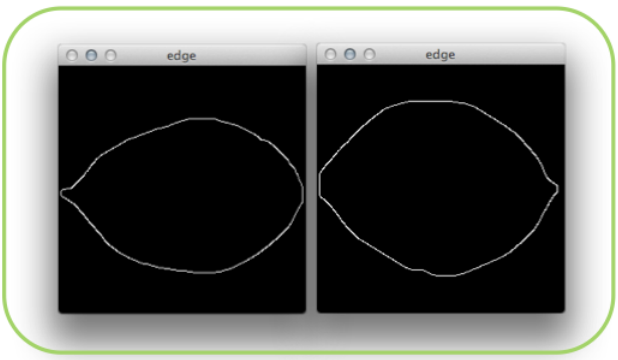


Fig. 5.4: Appropriate leaves contour

Subsequently, to solve the overlapping leaves as mentioned earlier, we utilize the Watershed method to discriminate the foreground leaf from its background leaf. As seen in figure 5.5, some degrees of overlapping leaves image can be perfectly segmented using the method that proposed in this research. All the sample overlapping leaf images from Bay, Cananga, Mangkokan, Jasmine, Cocor bebek, Vinca, Kestuba until Gardenia class are perfectly segmented.

However, this method still needs parameters as input to do erosion morphology operation to achieve the different markers. The parameters are type shape of Structuring Element (SE) as the base of morphology operation, and size of SE is must decided first. After decision of that parameters, we can easily do a segmentation process using watershed method. All the segmented leaf is the most upper leaf in the image.

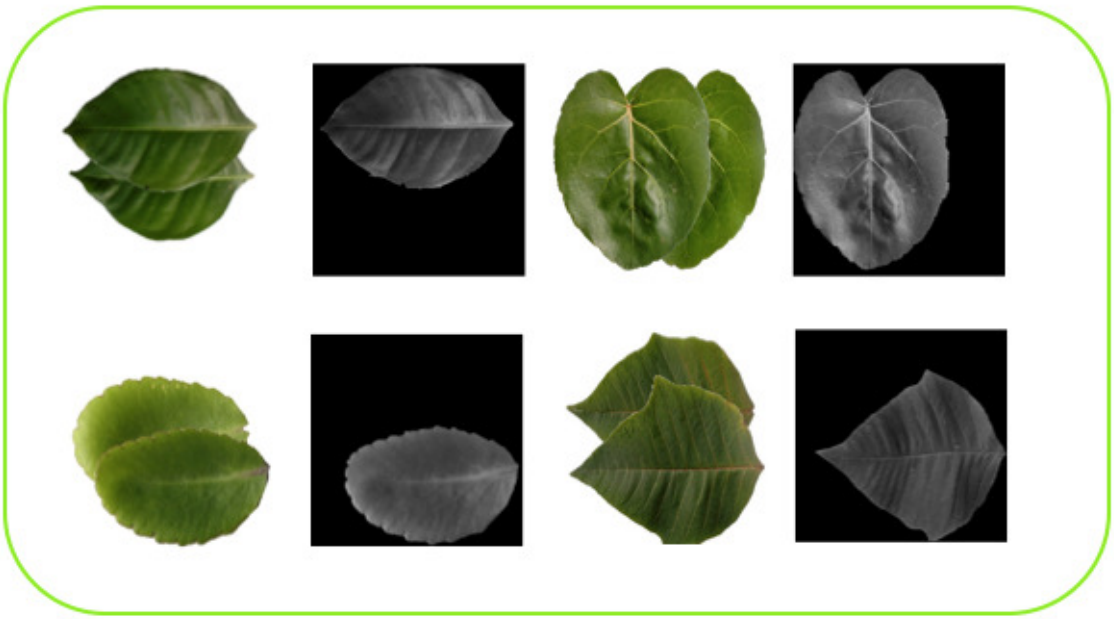


Fig. 5.5: Overlapping leaves segmentation results

5.2 Aspect checking

In order to check that this method is robust to translation, scale, rotation and Lighting aspects, we conduct the one-way ANOVA (Analysis of Variance) test. Any significant differences between two or more independent groups can be detected by using this test. This test discriminates through the variability of every value into a single component because of the variability among group means and other components because of the variability within the group. The ratio of two means square values called F-ratio. This test aspect will compare the F value with the F-criteria with two degrees of freedom (df) numbers.

A large F value in comparison with the F-criteria has meant that we need to reject the null hypothesis or the data between the group are different. The null hypothesis usually makes an assumption the data between the group is similar. Also, the small F-ratio have meant that we have to accept the null hypothesis or the data between the group are similar. Let denote the number of the group with k, and the number of its data element with n. The df_1 origins from the number of group minus 1 $(k - 1)$, where the df_2 is originally from the number of element data for all groups minus the number of groups $((n * k) - k)$.

5.2.1 Translation

The Dyadic wavelet transformation is only sampling the scaling parameter of the Continuous Wavelet Transform and made it robust to the translation. DT-CWT idea was inspired by the separation of the real part and imaginary part of the complex numbers. The scope of the translation aspect checking covers two degrees of translation, the small degree and big degree of translation. The image was translated to the left, upper, right and below parts of the image. In an example for left part have two degrees, translation left small degree and left big degree and presented in below figure.

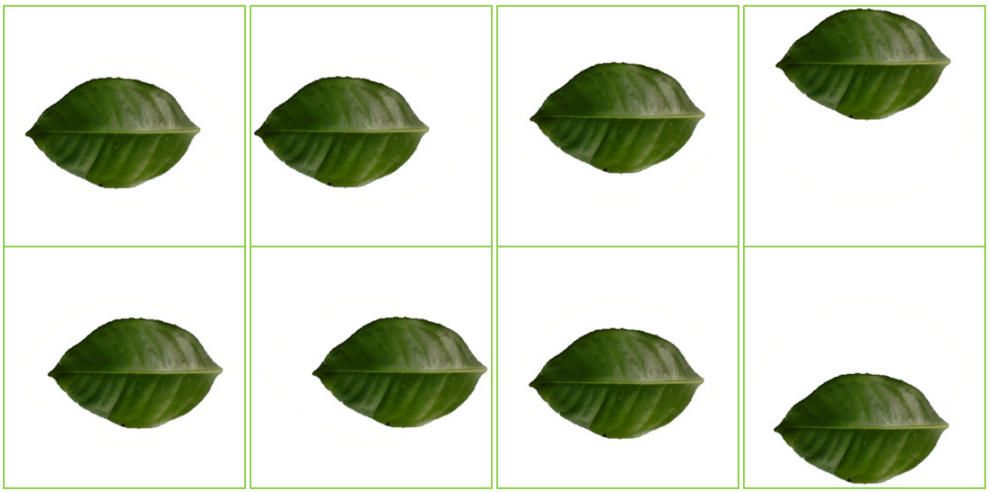


Fig. 5.6: Translated images

From table 5.1, we can see that the F values for all datasets are similar. Even though the F values are less than F criteria, but the Dyadic wavelet transform has showed us its ability in translation changes detection via its similar F values. On the other hand, the F values from the DT-CWT are very diverse. The important note from the table 5.2 comes from the right-big and below-big dataset. For these two datasets, the F values are greater than the F criteria, and it means that there are exist the different points among all images in the different group.

Table 5.1: ANOVA results for translation aspect (Dyadic)

Dataset	**F**	**F-crit**	**df_1**	**df_2**
left-big	0.1398	2.0868	7	120
left-small	0.1383			
upper-big	0.1392			
upper-small	0.1419			
right-big	0.1404			
right-small	0.1400			
below-big	0.1393			
below-small	0.1412			

Table 5.2: ANOVA results for translation aspect (DT-CWT)

Dataset	F	F-crit	df_1	df_2
left-big	1.2766	2.0339	7	376
left-small	0.8159			
upper-big	1.0968			
upper-small	1.5943			
right-big	2.1316			
right-small	1.2020			
below-big	2.2005			
below-small	0.6655			

5.2.2 Scale

The normalization step in the main process can guarantee that the wavelet transform results will have a strong ability in scale aspect. The image size was downscaled with different degrees to test this aspect, start from 10%, 20%, 30%, and 40%. This downscaling process was applied to images in every class.

Fig. 5.7: Downscaled images

Table 5.3: ANOVA results for scale aspect (Dyadic)

Dataset	**F**	**F-crit**	**df$_1$**	**df$_2$**
10%	99.0383	2.0118	7	4088
20%	19.7765			
30%	33.8348			
40%	55.8652			

As presented in table 5.3, the normalization step has contributed a superfine effort in scaling changes detection. All F values are greater than the F criteria raised to the null hypothesis rejection. The strongest representation of Dyadic result is presented in 10% of downscaled dataset. However, unlikely the Dyadic wavelet transform performance, the DT-CWT had the small F values for all downscaled datasets. All F values start from 10% dataset until 40% dataset are less than the F criteria value like showed in table 5.4.

Table 5.4: ANOVA results for scale aspect (DT-CWT)

Dataset	**F**	**F-crit**	**df$_1$**	**df$_2$**
10%	1.2004	2.0097	7	70024
20%	1.9518			
30%	0.5914			
40%	0.7253			

5.2.3 Rotation

Rotation is the important aspect to determine which method is an outperform method. Zernike complex moment is rotation-invariant method while the DT-CWT is relatively strong from rotation changes and the Dyadic wavelet transform also has an ability in rotation. The image was rotated in 22.5^0, 45^0, 67.5^0, 90^0, 112.5^0, 135^0, 157.5^0 and 180^0.

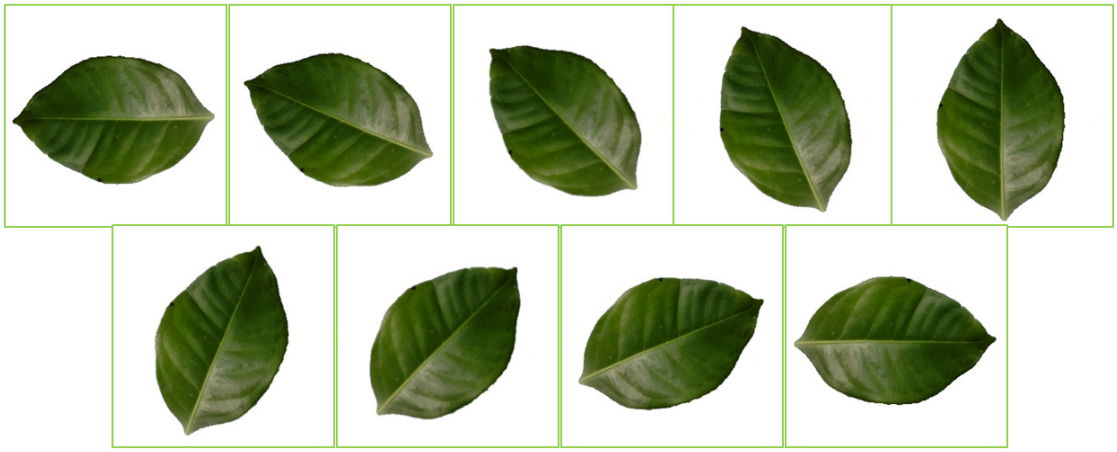


Fig. 5.8: Rotated images

Table 5.5: ANOVA results for rotation aspect (Dyadic)

Dataset (Degree)	**F**	**F-crit**	**df_1**	**df_2**
22.5	6.7117	2.0118	7	4088
45	1.7800			
67.5	3.5567			
90	5.9776			
112.5	3.5107			
135	1.3249			
157.5	5.8795			
180	10.3773			

Table 5.6: ANOVA results for rotation aspect (DT-CWT)

Dataset (Degree)	F	F-crit	df_1	df_2
22.5	1.5600	2.0097	7	81312
45	0.7544			
67.5	1.6885			
90	1.3047			
112.5	1.5667			
135	1.4802			
157.5	1.774			
180	1.3311			

In table 5.5 is presented the ANOVA test results toward the numbers of rotation changes datasets for Dyadic wavelet transform. Repeatedly, this transform gave the fine results for rotation changes detection from one class to another class. The absence of F values was found only in 45^0 and 135^0 while the remaining F values have great values. As presented in 5.6, the contrast results showed by the DT-CWT. This wavelet transform had the most ability only in 157.5^0 which showed by 1.774 of F value and still not reached the F criteria. Table 5.7 has a goal to determine the best number of low-order moments from Zernike moments method. It is found that the best representation comes from 35 moments' result.

Table 5.7: ANOVA results for rotation aspect (Zernike)

Dataset	F	F-crit	df_1	df_2
11 moments	0.0721	2.1263	7	80
27 moments	0.1289	2.0538	7	208
32 moments	0.1447	2.0466	7	248
35 moments	0.1662	2.0433	7	272

5.2.4 Lighting

Lighting is one of the big issues in this leaf identification system. The coverage of this test start from application of lighting impact in the left part, upper part, right part and below part of the leaf image. Short ranges of Hue, as well as Saturation, can help us in computation efficiency. From table 5.8 is presented the F values for several numbers of bin in color histogram calculation. This test was measured the data distribution among the images in the same class of the lighting changes. A very small F value informs us that the means of the entire groups are similar. The 24 bins' result has delivered a preferable result in comparison with the other.

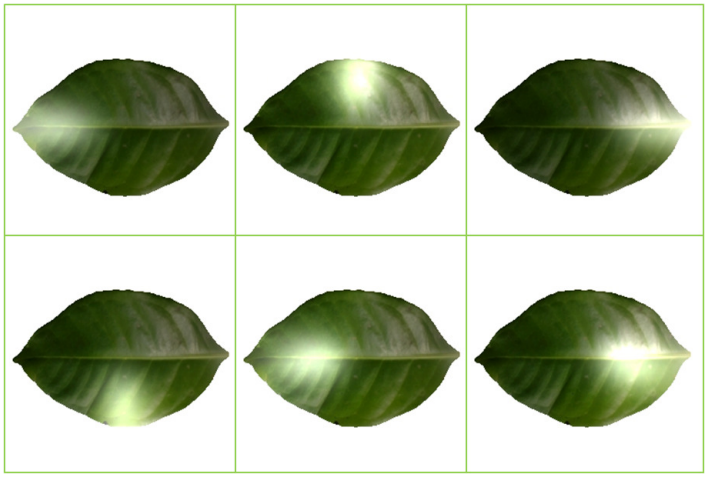


Fig. 5.9: Lighting changed images

Table 5.8: ANOVA results for lighting aspect

Dataset	**F**	**F-crit**	**df_1**	**df_2**
24 bins	1.95675E-05	2.1122	6	665
28 bins	2.02445E-05	2.1102	6	777
30 bins	1.97667E-05	2.1094	6	833
35 bins	2.03109E-05	2.1079	6	973

5.3 Classification result

5.3.1 Wavelet decomposition level

In this part, we want to investigate the best decomposition level for two wavelet transforms in this research. The Dyadic wavelet transform is a non-downsampling wavelet transform where the DT-CWT is downsampling wavelet transform with flexibility in the wavelet details. This different characteristic is motivated us to have knowledge of the better wavelet decomposition level. In the downsampling wavelet, the further level more than level 3 is usually the non-important information like noises, and in Dyadic wavelet transform also have a lack of information because of its vanishing moments filters. Aligned with that reason, we decided to use level 3 as an upper comparison level.

Table 5.9: Classification result for best texture extractor method determination

Dataset	**Dyadic**	**DT-CWT**
level 1	98.33	95.83
level 2	97.5	96.67
Level 3	96.67	99.17
Average	97.5	97.23

The values that demonstrated in table 5.9 are the classification results for each wavelet transform using designated decomposition level. Through experiment found that the key point in texture representation is located in the leaf venation. By using decomposition level 1 in Dyadic wavelet transform, leaf venation on the surface was detected very well. Similarly, using decomposition level 3 in DT-CWT, the texture of the leaf become more visible and easy to discriminate. For average performances of Dyadic wavelet transform and DT-CWT were obtained 97.5% and 97.23%, respectively. These results convinced us to consider Dyadic wavelet transform as texture extractor method.

5.3.2 Shape and texture methods

Classification is a further process to determine the best method that can represent leaf shape and texture very well. This classification process is only treated entire data in the entire class as the training set. Shape extractor methods consist of Zernike moments, Dyadic wavelet transforms, and DT-CWT. For each method has its characteristics. The Zernike moments has an ultimate advantage in rotation aspect while the Dyadic wavelet transform is non-downsampling wavelet transform that very effective in translation changes, and The DT-CWT presents the good ability in translation and rotation changes.

Table 5.10: Classification result for best shape extractor method determination

Dataset	**Zernike**	**Dyadic**	**DT-CWT**
Translation	69.17	100	95.83
Rotation	58.33	100	93.33
Scale	52.5	100	97.5

The results that presented in table 5.10 are training set results. The concept of training set result is to produce the classification model. This utilizes a dataset with known output values to build our model. The essential issue about the training set classification is the overfitting problem. This problem appears when we supply a lot of data into the model creation that affected to the model is perfectly created but only for that data. It was observed that as shape representation of the boundary-based had the positive results. The boundary-based methods are represented by the Dyadic wavelet transform and DT-CWT. The Zernike moments as a region-based shape extractor method didn't give satisfying result because of single main reason. The shape type of the leaf image inside database is dominantly occupied by the elliptical shape of the leaf such as Bay, Cananga, Jasmine and Gardenia. When we binarize this type of leaf shape, the shapes representation between these classes become identical.

Dyadic wavelet transform and DT-CWT are two methods in the texture features extraction. As previously mentioned, the texture of leaf shape is represented by using the energy, mean, standard deviation and coefficient values. Dyadic have three high-frequency components, and DT-CWT have six high-frequency components on both real tree and imaginary tree. This lead to different numbers of vector element between Dyadic and DT-CWT. Despite in limitation in high-frequency components, the Dyadic wavelet transform delivered us the satisfied results for translation, rotation and scale. The major different is located in rotation dataset. For the details please refer to table 5.11.

Table 5.11: Classification result for best texture extractor method determination

Dataset	**Dyadic**	**DT-CWT**
Translation	93.33	99.17
Rotation	90	100
Scale	95	99.17

5.3.3 Overall performance

The performances between the Dyadic wavelet and DT-CWT in aspect checking were nearly equal. This is motivated us to investigate the classification result for each feature towards the entire test aspects. Based on table 5.12 obtained the superfine results from the training set classification. It means that the classifier has successfully created the model for the data. In texture feature, the result was not perfect because of the number of texture feature elements was only 16 elements. This number can be categorized as a decent number while looking at the supplied test set results. The superfine result of shape feature in the training set tends to be overfitted the model, so when a new unknown data is inserted into the model, the model could not predict the output data very well. Dissimilarly, the test data in texture feature was perfectly matched the model and showed with the highest correct classification rate. The combination between three of the features was resulting a preferable result.

Table 5.12: Contribution per feature for Dyadic wavelet transform

Method	Shape	Color	Texture	All
Training set	100	100	94.17	100
Supplied test set	62.5	87.5	91.67	95.83

The error classification results in shape feature were occurred in Bay, Cananga, Mangkokan, Jasmine, and Kestuba class. Meanwhile, in color feature were occurred in Mangkokan, Jasmine, and Gardenia. The error prediction was also occurred for Jasmine class in texture feature and all features. The reason behind error classification for Jasmine class is because there are exist a wide range of different degree in shape, light and texture features.

Table 5.13: Classification results for several test aspects

Method	Original	Translation	Scaling	Rotation	Lighting	Average
Training set	100	100	100	100	100	**100**
Supplied test	95.83	87.5	91.67	87.5	83.33	**89.17**

The entire classification results in table 5.13 toward the training set were 100% and has meant that the classifier was able to produce the classification models. The overfitted problem will be easily known when a new unknown data is inserted into the model. If this problem occurs, the classification results the number of the testing set will be significantly decreased. However, as seen in that table, we gained the considerably fine of testing classification results. The highest percentage came from the original dataset where its images are not affected by any changes. The second place was occupied by scaling dataset with 91.67% of classification rate. Translation together with Rotation datasets have the same correct classification rates and were occupied the third and fourth places. Lighting datasets was occupied at the bottom that is the fifth place.

The error classification results in original dataset were from Jasmine class and for translation dataset was from Mangkokan, Jasmine, and Vinca. In scaling dataset, the error was found in Bay and Jasmine class while in lighting dataset found in Cananga, Jasmine, and Gardenia. Repeatedly, the highest error prediction in this measurement possessed by Jasmine class. It was observed that leaf venation on the leaf surface is the primary point of this ornamental leaf identification. Jasmine class has a wide range of different degree in leaf venation as well as lighting. Another problem is the different technique of leaf photo acquisition for all images in the database. However, based on the average of overall results, which is approximately about 90%, we can briefly make a statement that this leaf identification system is robust to translation, scaling, and lighting changes.

5.4 Enhancement in Color Feature Extraction

This additional work has an objective to enhance the color feature extraction towards sunlight as the dominant obstacle to outdoor image acquisition. The lighting impact from sunlight gives us a major negative effect in term of leaf identification. The different degrees of lighting impact can happen in the entire or partial part of leaf surface. This event leads the identification system to obtain false color information of the leaf image. In order to overcome this problem, we have to omit the aforementioned lighting impact as good as possible. One solution is the utilization of Infra Red (IR) camera rather than RGB or only or visible color camera. The IR has the longer wavelength, and it means that IR posses much wider information.

This work was conducted using a special web camera named Net Cowboy DC-NCR13U with the following specification:

- 4-layer glass lens
- Viewing angle 78 degrees
- Equipped with IR LED
- CMOS sensor with 1.3 million pixels resolution
- Support USB 2.0

Fig. 5.10: Identification tool used for color enhancement

This camera which visualized in image 5.10, allows RGB-color image acquisition as well as IR image acquisition. Because of distance limitation, the research object in this work was only used green plants collection inside Honjo Campus of Saga University. Single type of green plant were taken using the above camera both RGB-color and IR images. The sample of taken images are presented in image 5.11 and image 5.12.

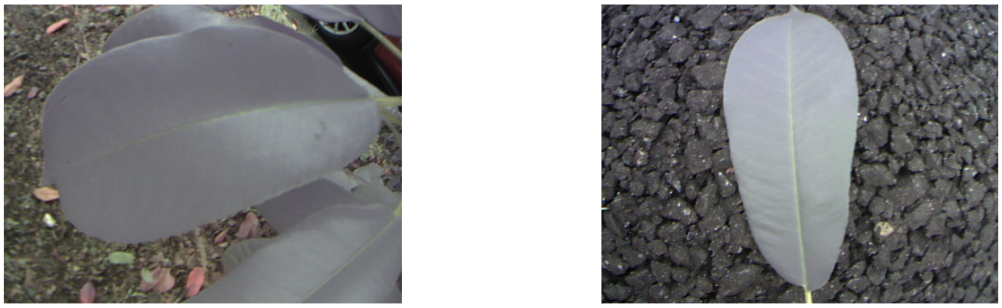
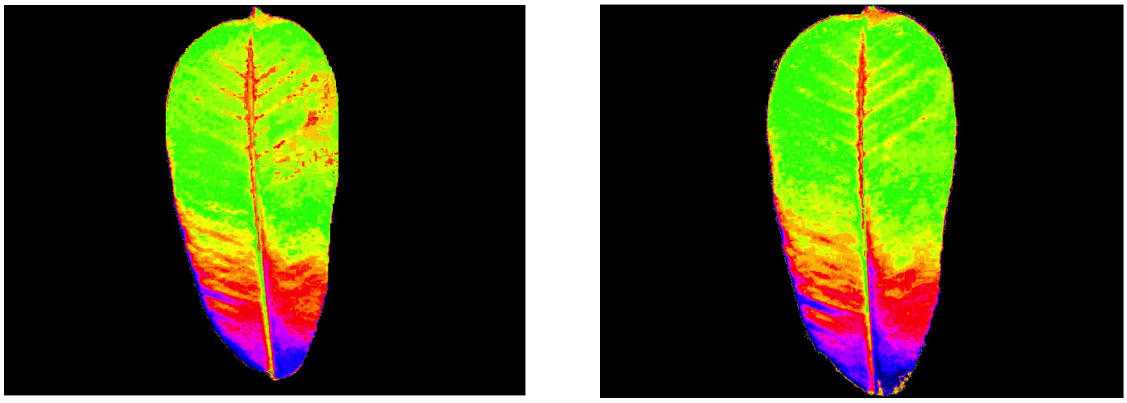


Fig. 5.11: RGB-color images

Fig. 5.12: IR images



(a) False-color of RGB image

(b) False-color of IR image

Fig. 5.13: False-color converted images

From the taken images, it is very hard to determine the different points between RGB-color and IR images. A special treatment to know this difference is needed. This treatment called False-color method. This method alters the initial color space to the more observable color space, assigning red color to the highest original pixel values and blue color to the lowest original pixel values. Orange, yellow and green as intermediate colors represent intermediate levels between red and blue colors. The image 5.13 is presented as the results.

From here, a small different point can be seen between two. On the upper-right side of the leaf, the IR image was able to reduce the lighting impact. Although the result was not significant, we can still make a conclusion that utilization of IR camera is expected to rise the correct classification rate of this identification process.

Chapter 6

System Implementation

In this section, we will explain the simple system implementation accordingly to the purpose of this research, which is to motivate or encourage people especially young people to know more about the ornamental leaf. The term of young people is closely related to technology, almost all young people nowadays have their own smart phone or tablet. This phenomenon motivates us to develop the application of as the implementation of this research.

The final application schema of this implementation is the application that can be used by two types of users. First is botanist or expert in plant as a contributor for the unknown plant. This person will identify the unknown plant based on his/her knowledge, and give the detail such as some photos of leaves, detail description of the leaf and medicinal function that can be obtained from the leaf. Second type of user is the ordinary user who only want to identify the known plant and get some information of the plant including its medicinal function. However, for this research we limit to the application only for the ordinary user. Please see figure 6.1 for reference.

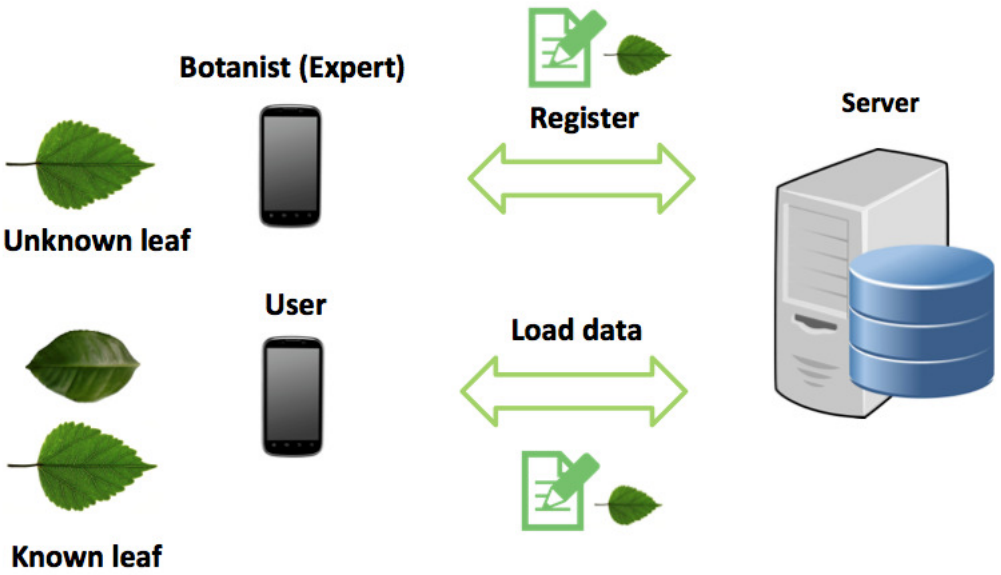


Fig. 6.1: Final application schema of this research

We choose Android operating system (OS) as implementation basis. The reason is that the market share for Android OS is much bigger in comparison with other OSs. Android OS is dominating with 78%, followed by iOS with 16%, Windows OS and BlackBerry OS with 3% and 2%, respectively. Another reason that motivates us to choose Android OS is because the price range of smartphone of Android loaded OS still affordable. Pie chart is displayed in the figure below:

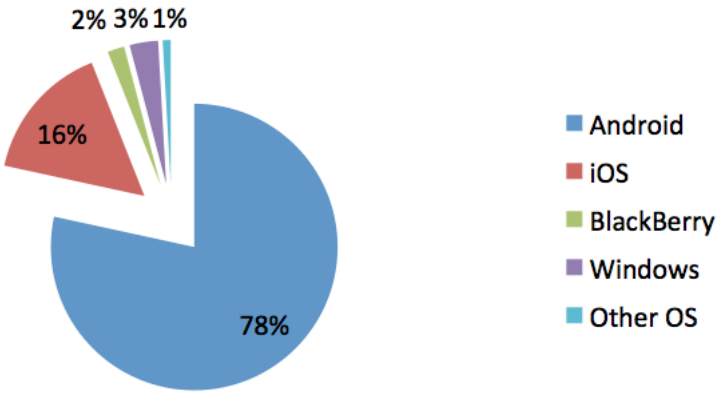


Fig. 6.2: Market share of smartphone worldwide in 2013

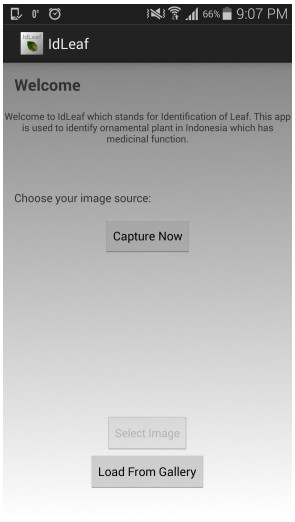
We have created a simple application for ordinary user as seen in figure 6.3 and 6.4. The user can choose the source of the image that want to be identified. This application offers two sources, first is Capture Now source and second is Load From Gallery source. Application flow as presented in fig 6.3 will be explained. At time the user opens the application, the splash screen will be displayed and continue to select source image page. From this page, the user will click the Load From Gallery button to load leaf image from the gallery, but before that user needs to select the gallery application. After the user selected the desired image, then the user can confirm through pressing the Select Image button. After the image is decided, then identification page will be displayed. The user can identify the image utilizing its color, shape, texture feature, or all three features. While the user pressing one of the provided buttons, it triggers result page to be displayed.

The application flow for Capture Now source is explained here. After the user open the application, splash screen is delivered to the front. It moves to select source file page, and from this point the user will choose the Capture Now button. The button triggers pop-up window to choose the most suitable camera application based on the user preferences. Camera application will take the image, then save to the gallery and showing the result to the confirm image page. After the user is sure about the image and clicks the Select Image button. Later, the identify image page will be shown, and the user can choose the feature that used for identification. However, because our limitation in the dataset, which is the plant collection in Indonesia, it can not fully implement yet. The result can be viewed in figure 6.4g.

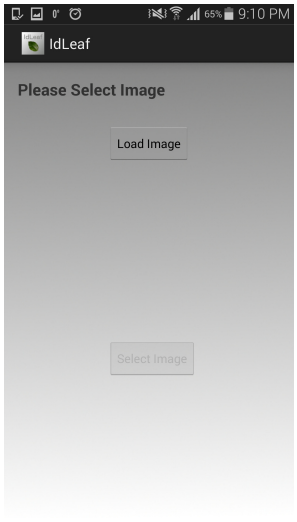
This application running in the following condition:

- Android OS 2.2 (Froyo) until 4.4 (Kitkat)
- API level 8 until API level 19
- Equipped with front and/or rear camera
- Installed SD card

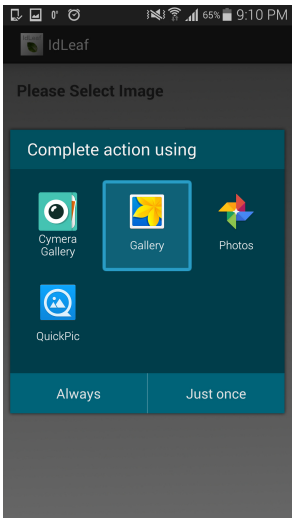
(a) Splash screen



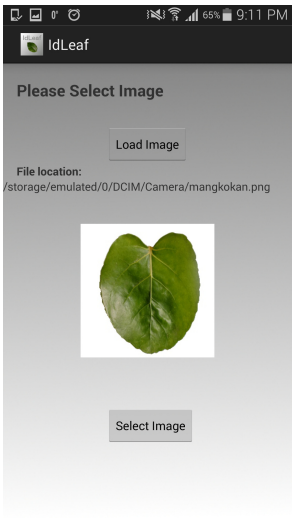
(b) Select source file



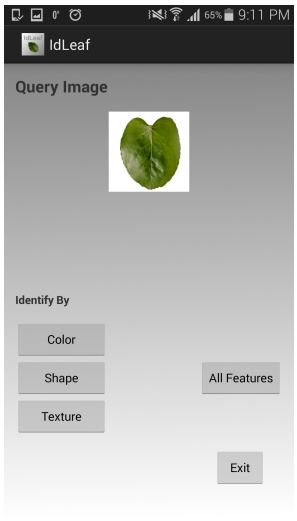
(c) Load from gallery



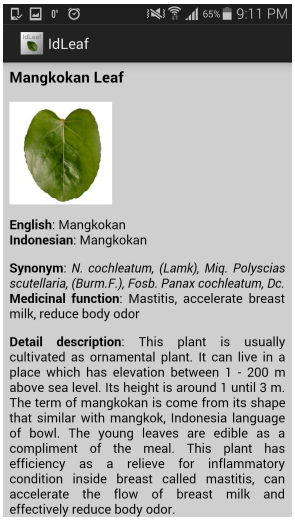
(d) Select gallery application



(e) Confirm image



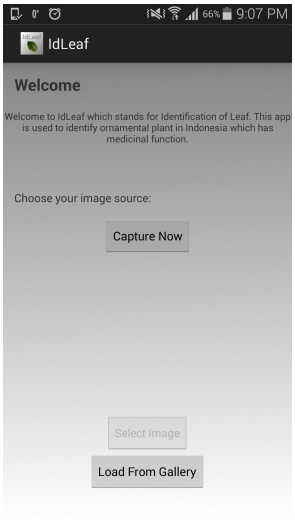
(f) Identify the image



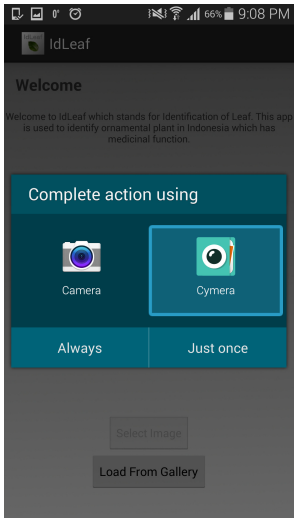
(g) Identification result

Fig. 6.3: Application flow for load image from gallery

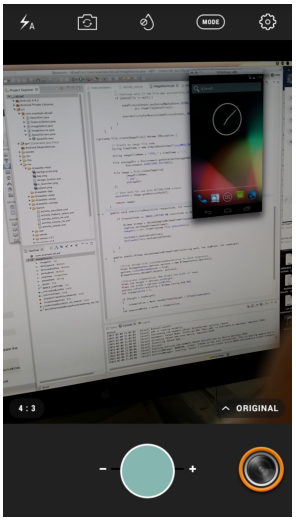
(a) Splash screen



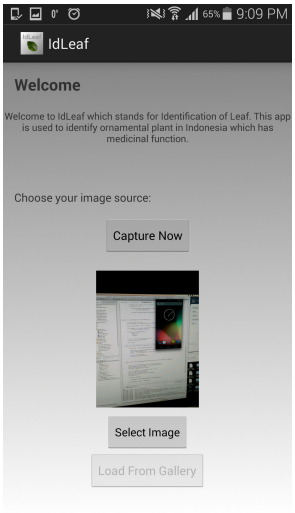
(b) Select source file



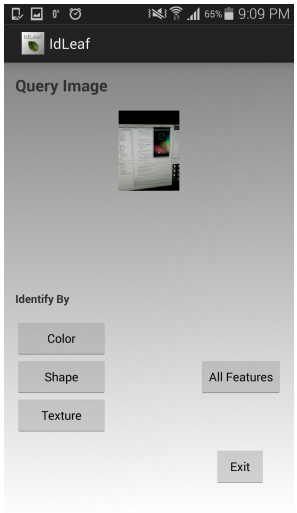
(c) Camera app selection



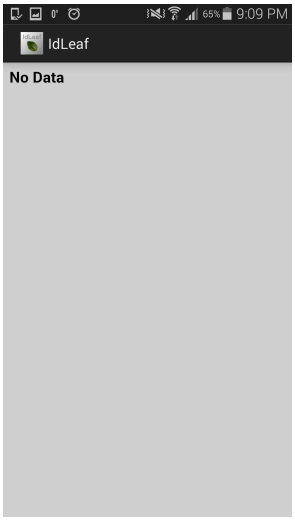
(d) Capture image



(e) Confirm image



(f) Identify the image



(g) Identification result

Fig. 6.4: Application flow for capture now

Chapter 7

Conclusion

This research has been conducted ornamental plant identification based on its leaf shape, texture and color features. For shape representation, we obtained from the experiments that the Dyadic wavelet transform was outperformed in comparison with the Zernike moments and DT-CWT. Although the wavelet transforms are not the perfect rotation-invariant methods, these methods have an ability also in rotation changes. As a texture feature extractor method, the Dyadic wavelet transform also gave us the highest results through its energy and statistical related values. In color extraction area, the HSV-based histogram had a good contribution to overcoming the lighting changes in leaf image.

Based on the results, we can conclude that this system is robust to translation, scaling, rotation and lighting changes, which are often happened in the real world situation. It was found that the best representation feature for ornamental leaf has an order that starts from its texture, followed by its color and then its shape. The Dyadic wavelet transform decomposed an ornamental leaf image from spatial space to spectrum space while maintaining its leaf venation. The observation results have showed us that leaf venation is the most important point to detect the leaf images inside the database. In addition, this research also can be extended to identify other types of plant.

Bibliography

[1] Numbers of threatened species by major groups of organisms 1996 - 2012 by iucn. http://www.iucnredlist.org/documents/summarystatistics/2012_2_RL_Stats_Table_1.pdf>. Accessed: 2013-06-21.

[2] T. Abdukirim, K. Niijima, and S. Takano. Lifting dyadic wavelet for denoising. Technical report, Department of Informatics, Kyushu University, Japan.

[3] Asa Ben-Hur and Jason Weston. *Data Mining Techniques for the Life Sciences*, volume 609, chapter A users guide to support vector machines, pages 223–239. Humana Press, 2010.

[4] Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines, 2013. http://www.csie.ntu.edu.tw/~cjlin/papers/libsvm.pdf.

[5] Ji-Xiang Du, Chuan-Min Zhai, and Qing-Ping Wang. Recognition of plant leaf image based on fractal dimension features. *Advanced Theory and Methodology in Intelligent Computing*, 116:150–156, 2013.

[6] Jan Flusser, Tomas Suk, and Barbara Zitova. *Moments and moments invariants in pattern recognition*. Wiley, United Kingdom, 1st edition, 2009.

[7] Rafael C. Gonzalez and Richard E. Woods. *Digital image processing*. Prentice Hall, 3rd edition, 2007.

[8] Sri Hartati. *Tanaman hias berkhasiat obat*. IPB Press, Bogor, Indonesia, 1st edition, 2011. In Indonesian.

[9] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. A practical guide to support vector classification, 2010. `http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf`.

[10] Stephane Mallat. *A wavelet tour of signal processing*. Academic Press, United States, 3rd edition, 2009.

[11] Jonathan Milgram, Mohamed Cheriet, and Robert Sabourin. One against one or one against all: which one is better for handwriting recognition with svms? In *Tenth International Workshop on Frontiers in Handwriting Recognition*. HAL archieve, 2006.

[12] Jin Kyu Park, Een Jun Hwang, and Yun Young Nam. Utilizing venation features for efficient leaf image retrieval. *Journal of Systems and Software*, 81(1):71–82, 2008.

[13] Greg Pass and Ramin Zabih. Comparing images using joint histograms, 1999.

[14] Chakravarti R. and Xiannong Meng. A study of color histogram based image retrieval. In *2009 Sixth International Conference on Information Technology: New Generations*, pages 1323–1328. IEEE, 2009.

[15] Ivan W. Selesnick, Richard G. Baraniuk, and Nick G. Kingsbury. The dual-tree complex wavelet transform. *Signal Processing Magazine*, 22:123–151, 2005.

[16] Jamie Shutler. Complex zernike moments, 2002. `http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/SHUTLER3/node11.html`.

[17] Jean-Luc Starck, Fionn Murtagh, and Jalal M. Fadli. *Sparse image and signal processing*. Cambridge University Press, United States, 1st edition, 2010.

[18] Satoshi Suzuki and Keiichi Abe. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30:32–46, 1985.

[19] Amir Tahmasbi, Fatemeh Saki, and Shariar B. Shokouhi. Classification of benign and malignant masses based on zernike moments. *Computer in Biology and Medicine*, 41:726–735, 2011.

[20] Xiao-Feng Wang, De-Shuang Huang, Ji-Xiang Du, Xuan Xu, and Laurent Heutte. Classification of plant leaf images with complicated background. *Applied Mathematics and Computation*, 205(2):916–926, 2008.

[21] Xuan Wang, Junhua Liang, and Fangxia Guo. Feature extraction algorithm based on dual-scale decomposition and local binary descriptors for plant leaf recognition. *Digital Signal Processing*, 34:101–107, 2014.

[22] Zhiyong Wang, Zheru Chi, Dagan Feng, and Qing Wang. Leaf image retrieval with shape features. *Advances in Visual Information Systems*, 1929:477–487, 2000.

[23] Herlina Widyaningrum. *Kitab tanaman obat*. Media Pressindo, Jogjakarta, Indonesia, 1st edition, 2011. In Indonesian.

[24] Ian H. Witten and Eibe Frank. *Data mining*. Morgan Kaufmann Press, United States, 2nd edition, 2005.

[25] Xiang-Qian Wu, Kuan-Quan Wang, and David Zhang. Wavelet energy feature extraction and matching for palmprint recognition. *Journal of Computer Science and Technology*, 20:411–418, 2005.

[26] Dengsheng Zhang and Guojun Lu. A comparative study of fourier descriptors for shape representation and retrieval. In *Proc. of 5th Asian Conference on Computer Vision (ACCV)*, pages 646–651. Springer, 2002.

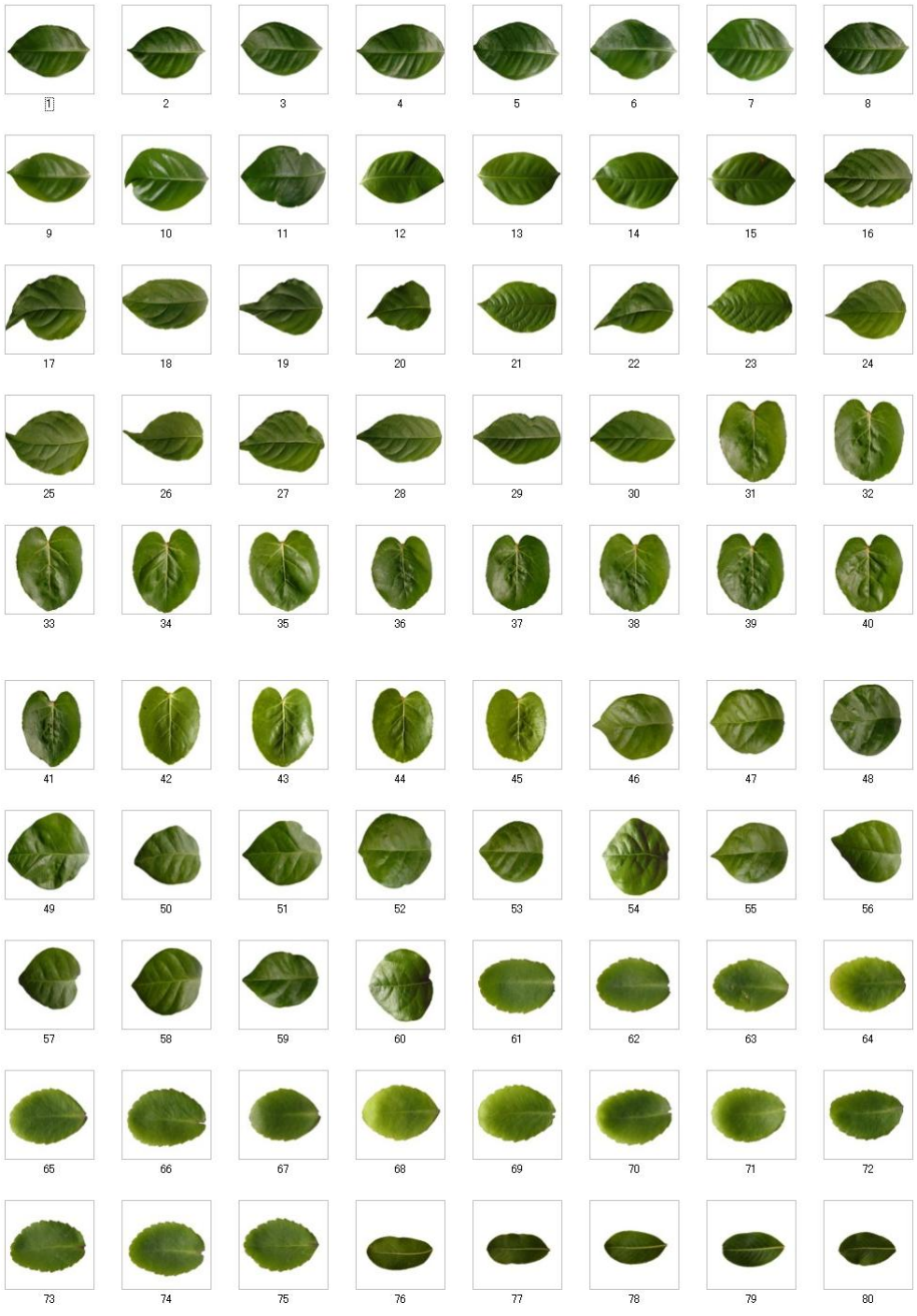
Acknowledgements

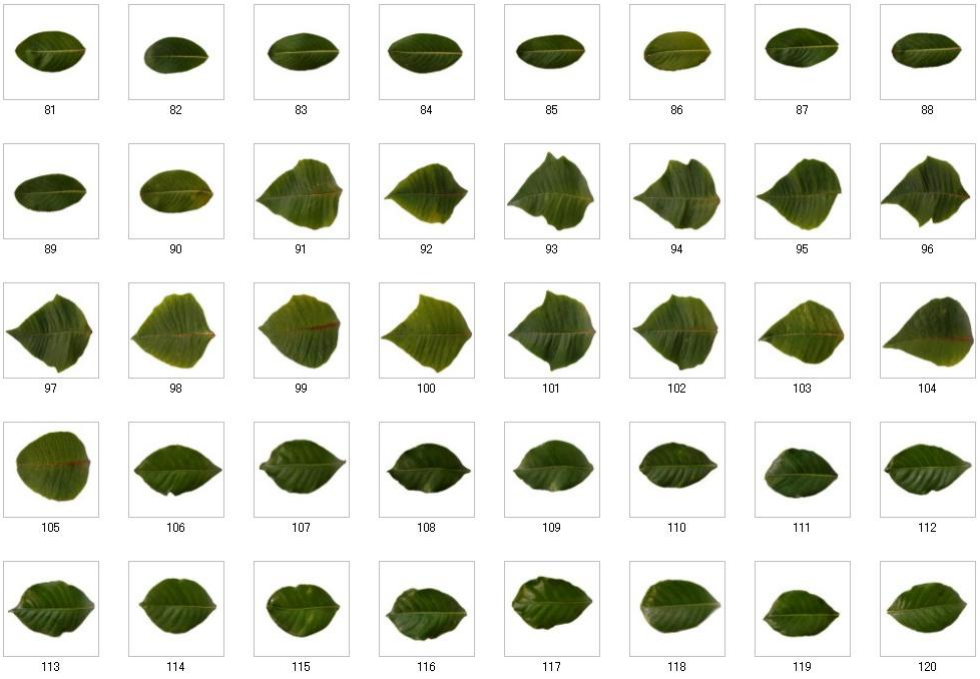
First of all I would like to express my gratitude because destined to have the great supervisors like Prof. Kohei Arai and Prof. Hiroshi Okumura. During my study in this degree, they are continuously supporting me through their knowledge and love. Regarding their outstanding ability, I obtained much information and knowledges that leverage my skills especially in image processing and remote sensing areas. Then, certainly I will not be graduated if there is no will from Allah SWT, be grateful and happy of this blessing. Moreover, I liked to deliver thanks to my Father and Mother in Indonesia as well as the most precious persons in my life, my lovely wife, Nila Yantrisiana Puspitasari and my lovely son, Ryouta Altaf Ghaisan. Their infinite supports and loves made me possible to get this degree. Also many thanks to my Sister and Brother in law for their effort that have opened a way to start the study at Saga University. Last but not least many thanks to all my lab mates. They were remarkably helped me to keep struggling with my research.

Appendix A

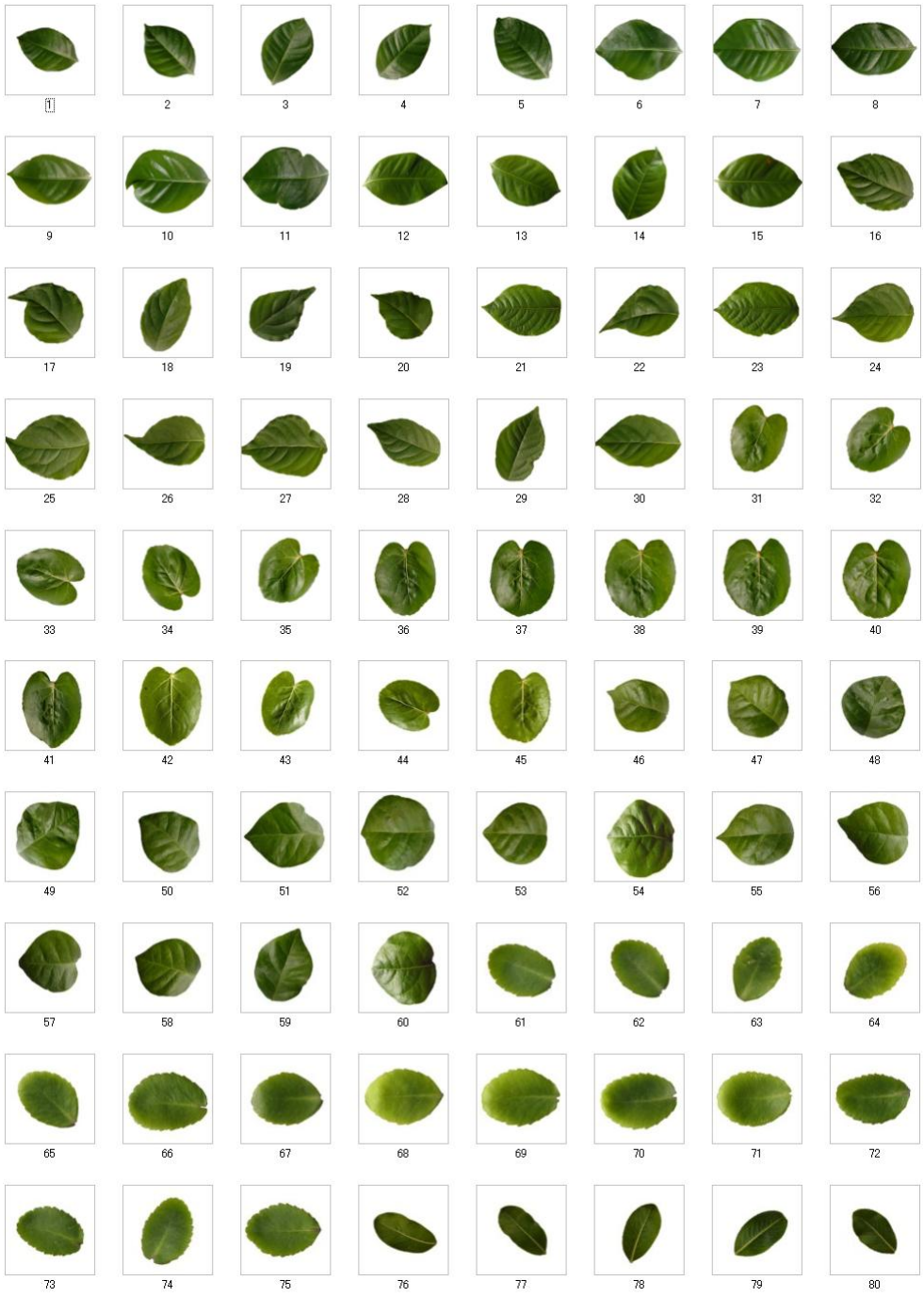
Image Dataset

Original


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80

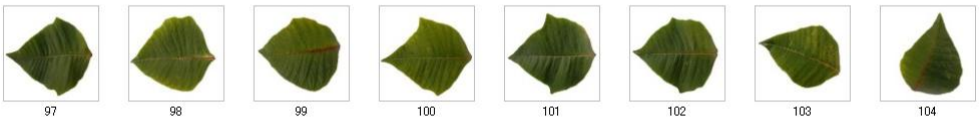

81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120

Rotate


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80


81
82
83
84
85
86
87
88

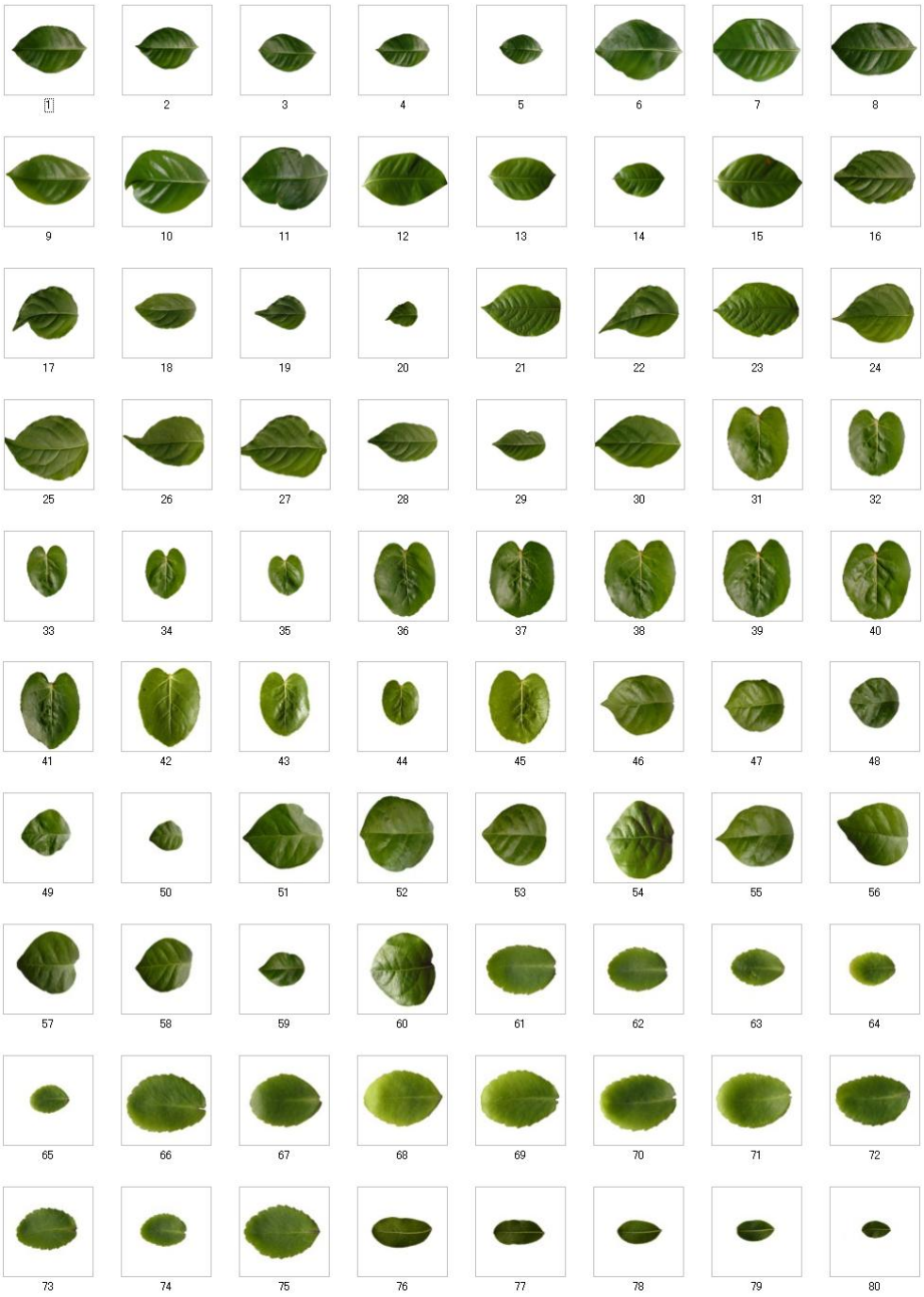

89
90
91
92
93
94
95
96


97
98
99
100
101
102
103
104


105
106
107
108
109
110
111
112

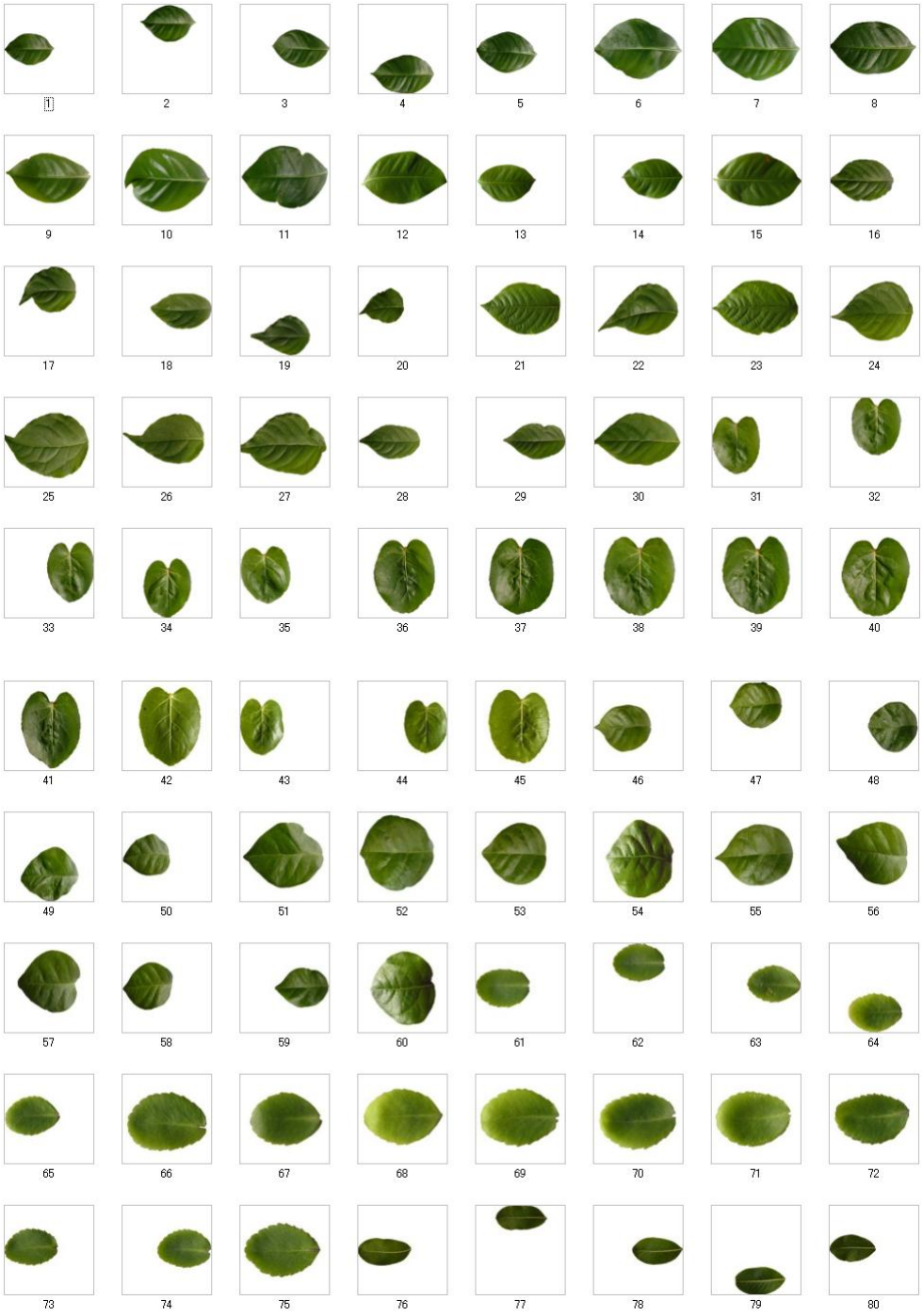

113
114
115
116
117
118
119
120

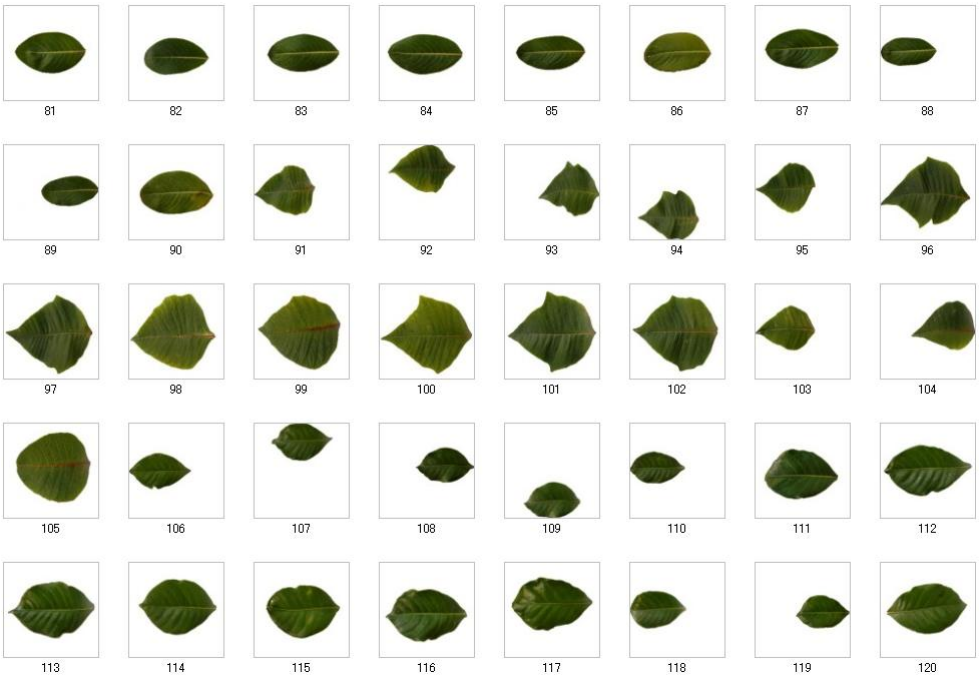
Scale


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80


81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120

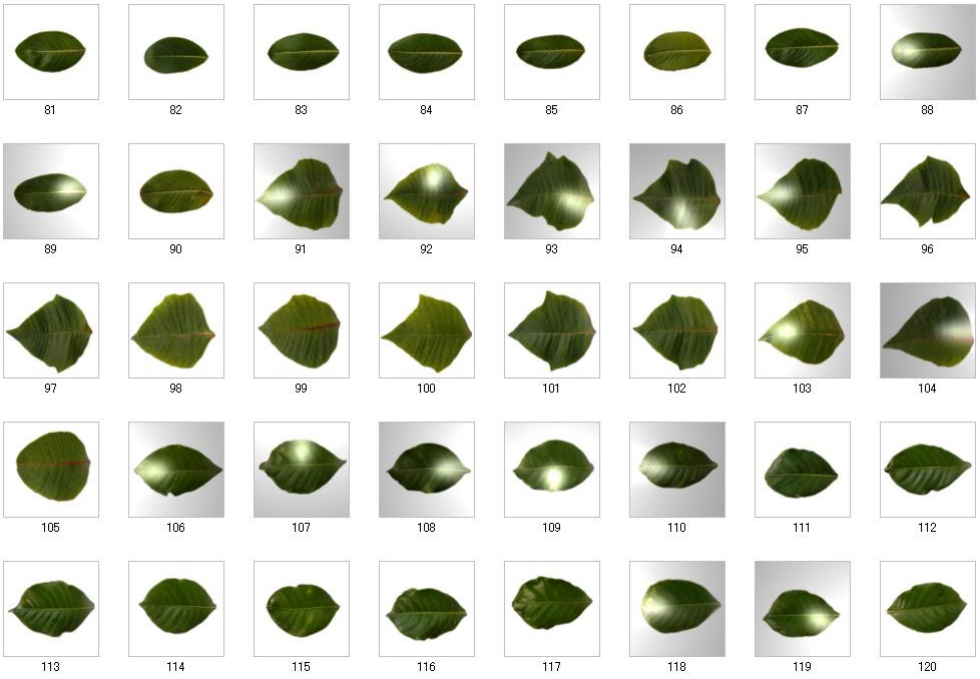
Translate


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80


81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120

Lighting


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80


81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120

Appendix B

Code

```
#include <iostream>
#include <string>
#include <fstream>
#include <sstream>

#ifdef __cplusplus
#include <opencv2/opencv.hpp>
#endif

//size of input image is 256
#define SIZE_ORI 256
//size for dyadic decomposed image is 256 x 2
#define SIZE SIZE_ORI*2

//number image in database is 120
#define NUM_IM 120

//number of texture feature for each detail is 4
#define NUM_DESC_TEX 4
//if all details is used then 4 x 4
#define NUM_DESC_TEXTURE NUM_DESC_TEX*4

//number of shape descriptors
#define NUM_BOUNDARY_PXL 1200
//number of shape descriptor after feature reduction using PCA
//and for each detail
#define NUM_DESC_SHAPE 50

//number of bins used in color histo per region
#define NUM_BINS 24
//we divided 4 regions so all bins is 24 x 4
#define NUM_DESC_COLOR NUM_BINS*4

using namespace cv;
using namespace std;

void preProcess (Mat input, Mat& output);
void otsuThreshold (Mat input);

void writeContour(Mat input, Mat contour_output);
void measureDist (Mat input, double mat[NUM_BOUNDARY_PXL * 2], Point
```

```
    centroid, Mat maxval, int index);
double euclidianDist (Point p, Point q);
void dyadic_1 (double mat[NUM_BOUNDARY_PXL * 2], int n);
void dyadic_1d(double mat[NUM_BOUNDARY_PXL * 2], int level);
void copyToMat (double matrix[NUM_BOUNDARY_PXL * 2], int index, int
    subband, Mat output);
void scaleInvariant(Mat& input, Mat maxval);
void dimensionReduction (Mat input, Mat& output);

void dyadicDescTexture (Mat input, int index, int level, int region, Mat
    desc);
void dyadic_2d(double mat[][SIZE], int quarter);
void dyadic_2 (double mat[][SIZE], int n, int row);
void transpose (double matrix[][SIZE], int quarter);
void copyToArr_2d (Mat input, double matrix[][SIZE]);
void waveDetailDyad (Mat mat, Mat& output, int detail);
void waveletDesc (Mat input, int index, int index_desc, Mat& output);

void calc_histo(Mat subimage, int feature_num, int index, Mat color_desc);
void subImage(Mat src, int spoint, int epoint, int size, int index, Mat
    color_desc);

void appendAllDesc (Mat shp1, Mat shp2, Mat tex, Mat col, Mat output);

int main(int argc, const char * argv[])
{
    //creating variable to store input folder location
    string file_loc = "Classification/scale/";

    //creating file_output to save final results
    ofstream file_output ("all_scale.csv", ofstream::trunc);

    //checking file whether opened or not
    if (!file_output.is_open()){
        cout << "File not opened" << endl;
        return 1;
    }
    //variable to store low freq comp of shape descriptor
    Mat output_desc_shp_low = Mat::zeros (NUM_IM, NUM_BOUNDARY_PXL,
    CV_64FC1);
    //variable to store high freq comp of shape descriptor
    Mat output_desc_shp_high = Mat::zeros (NUM_IM, NUM_BOUNDARY_PXL,
    CV_64FC1);
    //variable to store texture features
    Mat output_desc_tex = Mat::zeros (NUM_IM, NUM_DESC_TEXTURE, CV_64FC1);
    //variable to store color features
    Mat output_desc_col = Mat::zeros (NUM_IM, NUM_DESC_COLOR, CV_64FC1);
    //variable to store farthest dist for every image
    Mat  maxval = Mat (1, NUM_IM, CV_16UC1);

    //looping process for all images inside database
    for (int index = 1; index <= NUM_IM; index++){

        //variables to store input image location
        string file_loc_im;
        stringstream image_number;
```

```
92
93          //creating opencv mat files
94          Mat input_image, input_post, input_post_col, contour_output;
95          //creating variables to store shape descriptor in ordinary C array
96          double desc_shp[NUM_BOUNDARY_PXL * 2] = {0};
97
98          //pass index to image number
99          image_number << index;
100
101         //append folder name and image number
102         file_loc_im = file_loc + image_number.str() + ".jpg";
103         cout << "processing image number " << image_number.str() << endl;
104
105         //reading input image
106         input_image = imread(file_loc_im, CV_LOAD_IMAGE_COLOR);
107
108         //if there is no specified input image then exit
109         if (input_image.empty()) {
110             cout << "No Image is loaded" << endl;
111             return 1;
112         }
113
114         //resize(input_image, input_image, Size (SIZE_ORI, SIZE_ORI));
115
116         //creating var to store preprocessing result in gray
117         input_post = Mat (input_image.rows, input_image.cols, CV_8UC1);
118
119         //creating var to store preprocessing result in color
120         input_post_col = Mat (input_image.rows, input_image.cols, CV_8UC3)
    ;
121
122         //obtain perfect shape of leaf for gray image
123         preProcess(input_image, input_post);
124
125         //obtain perfect shape of leaf for color image
126         preProcess(input_image, input_post_col);
127
128         //level is wavelet decomposition level
129         //region is region to be used for texture description
130         //region equal with 0 means approximation region only
131         //region equal with 1 means high regions only
132         //region equal with 2 means all region
133         int level = 1; int region = 2;
134
135         //SHAPE feature extraction
136         //creating zeros-valued matrix to store leaf contour
137         contour_output = Mat::zeros(SIZE_ORI, SIZE_ORI, CV_8UC1);
138
139         //detect the contour
140         writeContour(input_post, contour_output);
141
142         //find leaf image's centroid
143         Moments mmt = moments(contour_output);
144         Point centro = Point (mmt.m10/mmt.m00, mmt.m01/mmt.m00);
145
146         //measure contour pixels with its centroid
```

```
147         measureDist(contour_output, desc_shp, centro, maxval, index);
148 
149         //perform dyadic wavelet transform
150         dyadic_1d(desc_shp, level);
151 
152         //subband option to choose the low or high details
153         //because 1D it means we only have have 1 low and 1 high regions
154         //1 is low and 2 is high
155         int subband = 1;
156         copyToMat(desc_shp, index, subband, output_desc_shp_low);
157 
158         //subband option to choose the low or high details
159         //1 is low and 2 is high
160         subband = 2;
161         copyToMat(desc_shp, index, subband, output_desc_shp_high);
162         //SHAPE feature extraction
163 
164         //TEXTURE feature extraction
165         //calculate statistical-related features from each detail
166         dyadicDescTexture(input_post, index, level, region,
    output_desc_tex);
167         //TEXTURE feature extraction
168 
169         //COLOR feature extraction
170         //calculate HSV color histogram
171         int xpoint = 0, ypoint = 0, roisize = (SIZE_ORI/2);
172 
173         subImage(input_post_col, xpoint, ypoint, roisize, index,
    output_desc_col);
174         //COLOR feature extraction
175     }
176 
177     //SHAPE feature reduction using PCA
178     Mat output_reduction_shp_low, output_reduction_shp_high, all_desc;
179 
180     //make shape descriptors invariant to scale
181     scaleInvariant(output_desc_shp_low, maxval);
182     scaleInvariant(output_desc_shp_high, maxval);
183 
184     output_reduction_shp_low = Mat (NUM_IM, NUM_DESC_SHAPE, CV_64FC1);
185     output_reduction_shp_high = Mat (NUM_IM, NUM_DESC_SHAPE, CV_64FC1);
186 
187     //dimension reduction using PCA
188     dimensionReduction(output_desc_shp_low, output_reduction_shp_low);
189     dimensionReduction(output_desc_shp_high, output_reduction_shp_high);
190     //Shape feature reduction using PCA
191 
192     //creating var to store all features
193     all_desc = Mat::zeros(NUM_IM, (NUM_DESC_SHAPE*2) + NUM_DESC_TEXTURE +
    NUM_DESC_COLOR, CV_64FC1);
194 
195     //append all features file become one file
196     appendAllDesc(output_reduction_shp_low, output_reduction_shp_high,
    output_desc_tex, output_desc_col, all_desc);
197 
198     //print results as CSV file
```

```
199     file_output << format(all_desc, "CSV") << endl;
200 
201     //close file
202     file_output.close();
203 
204     return 0;
205 }
206 
207 //To change the mat in function, we need to define as parameter-reference
    (&)
208 //Preprocessing input image from any noises
209 void preProcess (Mat input, Mat& output){
210 
211     Mat input_gray, input_thresh, s_element;
212 
213     cvtColor(input, input_gray, CV_BGR2GRAY);
214 
215     input_gray.copyTo(input_thresh);
216 
217     otsuThreshold(input_thresh);
218 
219     threshold(input_thresh, input_thresh, 250, 255, THRESH_BINARY);
220 
221     s_element = getStructuringElement(MORPH_ELLIPSE, Size (3,3), Point
    (1,1));
222 
223     morphologyEx(input_thresh, input_thresh, MORPH_OPEN, s_element);
224 
225     //if function parameter for output has 1 channel
226     //then mantain the channel after preprocessed
227     if (output.channels()==1){
228 
229         input_gray.copyTo(output);
230 
231         //in opencv binary image 0 is black
232         for (int i = 0; i < input.rows; i++){
233             for (int j = 0; j < input.cols; j++){
234 
235                 if (input_thresh.at<uchar>(i,j) == 255){
236                     output.at<uchar>(i,j) = 0;
237                 }
238             }
239         }
240 
241     }
242 
243     //if function parameter for output has 3 channel
244     //then mantain the channel after preprocessed
245     else if (output.channels()==3){
246 
247         input.copyTo(output);
248 
249         for (int i = 0; i < input.rows; i++){
250             for (int j = 0; j < input.cols; j++){
251 
252                 if (input_thresh.at<uchar>(i,j) == 255){
```

```
                    output.at<Vec3b>(i,j)[0] = 255;
                    output.at<Vec3b>(i,j)[1] = 255;
                    output.at<Vec3b>(i,j)[2] = 255;
                }
            }
        }
    }
}

//Otsu thresholding to unnecessary region
void otsuThreshold (Mat input){

    threshold(input, input, 0, 255, CV_THRESH_BINARY|CV_THRESH_OTSU);

}

//detect leaf contour in input image
void writeContour(Mat input, Mat contour_output){

    Mat edge, input_copy;
    vector< vector<Point> > contours;
    vector< Vec4i > hierarchy;

    double largest_area = 0.0;
    int largest_contour_index = 0, canny_thresh;
    Rect bounding_rect;

    input.copyTo(input_copy);

    //edge detection
    canny_thresh = 200;

    Canny(input_copy, edge, canny_thresh, canny_thresh*3);

    //find leaf contour
    findContours(input_copy, contours, hierarchy, CV_RETR_TREE,
   CV_CHAIN_APPROX_SIMPLE, Point(0,0));

    //select biggest contour based on contour area
    //iterate through each contour.
    for( int i = 0; i < contours.size(); i++) {

        //find the area of contour
        double a = contourArea(contours[i], false);
        if(a > largest_area){

            largest_area = a;
            //store the index of largest contour
            largest_contour_index = i;
            //find the bounding rectangle for biggest contour
            bounding_rect = boundingRect(contours[i]);
        }
    }

    Mat drawing = Mat::zeros(input.size(), CV_8UC1);

```

```
308     //drawing only largest contour
309     Scalar color = Scalar(255);
310 
311     drawContours(drawing, contours, largest_contour_index, color, 1, 8,
        hierarchy, 0, Point());
312 
313     for (int i = 0; i < input.rows; i++)
314         for (int j = 0; j < input.cols; j++){
315 
316             if (drawing.at<uchar>(i,j) == 255){
317 
318                 contour_output.at<uchar>(i,j) = 255;
319             }
320         }
321 }
322 
323 //distance calculation between contour pixels and centroid
324 void measureDist (Mat input, double mat[NUM_BOUNDARY_PXL * 2], Point
        centroid, Mat maxval, int index){
325 
326     int k = 0;
327     double max = DBL_MIN;
328     int max_point = 0;
329 
330     for (int i = 0; i < input.rows; i++)
331         for (int j = 0; j < input.cols; j++){
332 
333             if (input.at<uchar>(i,j) > 0){
334                 mat[k] = euclidianDist(Point (i,j), centroid);
335 
336                 if (mat[k] > max){
337                     max = mat[k];
338                     max_point = k;
339                 }
340                 k++;
341 
342             }
343         }
344 
345     maxval.at<int>(0,index-1) = max_point;
346 
347 }
348 
349 //calculate euclidian distance between two points
350 double euclidianDist (Point p, Point q){
351 
352     double dist = norm(p-q);
353 
354     return dist;
355 }
356 
357 //1D dyadic wavelet transform looping
358 void dyadic_1d(double mat[NUM_BOUNDARY_PXL * 2], int level){
359 
360     for (int i = 0; i < level; i++){
361         dyadic_1 (mat, NUM_BOUNDARY_PXL * 2);
```

```
362     }
363
364 }
365
366 //1D dyadic wavelet transform
367 void dyadic_1 (double mat[NUM_BOUNDARY_PXL * 2], int n){
368
369     double *tempbank = 0;
370     double h1, h2, h3, h4, g1, g2, g3, g4, g5, g6;
371     int i, j;
372
373     j = 1;
374
375     if (tempbank == 0)
376         tempbank = (double *)malloc(n * sizeof (double));
377
378     for (i = 0; i < n; i++)
379         *(tempbank+i) = 0;
380
381
382     h1 = 0.125*sqrt(2.0);
383     h2 = 0.375*sqrt(2.0);
384     h3 = 0.375*sqrt(2.0);
385     h4 = 0.125*sqrt(2.0);
386
387     g1 = -0.03125*sqrt(2.0);
388     g2 = -0.09375*sqrt(2.0);
389     g3 = -0.5625*sqrt(2.0);
390     g4 = 0.5635*sqrt(2.0);
391     g5 = 0.09375*sqrt(2.0);
392     g6 = 0.03125*sqrt(2.0);
393
394
395     for (i = 1; i < (n/2)-2; i++){
396
397         if (i == 1){
398             tempbank[j*2] = (g1 * mat[n-1]) + (g2 * mat[i-1]) + (g3 * mat[
    i])+ (g4 * mat[i+1])+ (g5 * mat[i+2])+ (g6 * mat[i+3]);
399             tempbank[j*2+1] = (h1 * mat[i-1]) + (h2 * mat[i]) + (h3 * mat[
    i+1])+ (h4 * mat[i+2]);
400         }
401
402         else if (i == ((n/2)-3)){
403             tempbank[j*2] = (g1 * mat[i-2]) + (g2 * mat[i-1]) + (g3 * mat[
    i])+ (g4 * mat[i+1])+ (g5 * mat[i+2])+ (g6 * mat[0]);
404             tempbank[j*2+1] = (h1 * mat[i-1]) + (h2 * mat[i]) + (h3 * mat[
    i+1])+ (h4 * mat[i+2]);
405         }
406
407         else{
408             tempbank[j*2] = (g1 * mat[i-2]) + (g2 * mat[i-1]) + (g3 * mat[
    i])+ (g4 * mat[i+1])+ (g5 * mat[i+2])+ (g6 * mat[i+3]);
409             tempbank[j*2+1] = (h1 * mat[i-1]) + (h2 * mat[i]) + (h3 * mat[
    i+1])+ (h4 * mat[i+2]);
410         }
411
```

```
412         j++;
413     }
414 
415     //handling edge problem because of number of filters used
416     //front edge problem
417     tempbank[0] = (g1 * mat[(n/2)-2]) + (g2 * mat[(n/2)-1]) + (g3 * mat
        [0])+ (g4 * mat[1])+ (g5 * mat[2])+ (g6 * mat[3]);
418     tempbank[1] = (h1 * mat[(n/2)-1]) + (h2 * mat[0]) + (h3 * mat[1])+ (h4
         * mat[2]);
419 
420     //end edge problem
421     tempbank[n-2] = (g1 * mat[(n/2)-4]) + (g2 * mat[(n/2)-3]) + (g3 * mat
        [(n/2)-2])+ (g4 * mat[(n/2)-1])+ (g5 * mat[0])+ (g6 * mat[1]);
422     tempbank[n-1] = (h1 * mat[(n/2)-3]) + (h2 * mat[(n/2)-2]) + (h3 * mat
        [(n/2)-1])+ (h4 * mat[0]);
423 
424 
425     for (i = 0; i < n; i++){
426         if (i%2==0){
427             mat[n/2 + i/2] = tempbank[i];
428         }
429         else{
430             mat[i/2] = tempbank[i];
431         }
432     }
433 
434     delete tempbank;
435 }
436 
437 //copy transformed data according to detail
438 void copyToMat (double matrix[NUM_BOUNDARY_PXL * 2], int index, int
        subband, Mat output){
439 
440     //default is high detail
441     int j = NUM_BOUNDARY_PXL;
442 
443     if (subband == 1)
444         j = 0;
445     else if (subband == 2)
446         j = NUM_BOUNDARY_PXL;
447 
448     for (int i = 0; i < NUM_BOUNDARY_PXL; i++){
449         output.at<double>(index-1,i) = matrix[j];
450         j++;
451     }
452 
453 }
454 
455 //make shape descriptor is invariant to scale
456 void scaleInvariant(Mat& input, Mat maxval){
457 
458     int i, j;
459 
460     for (i = 0; i < input.rows; i++){
461         for (j = 0; j < input.cols; j++){
462             input.at<double>(i,j) /= input.at<double>(i,maxval.at<int>(0,i
```

```
    ));
463         }
464     }
465
466 }
467
468 //feature reduction using PCA
469 void dimensionReduction (Mat input, Mat& output){
470
471     PCA pca (input, Mat(), CV_PCA_DATA_AS_ROW, NUM_DESC_SHAPE);
472
473     pca.project(input, output);
474
475 }
476
477
478 //calculate texture feature
479 void dyadicDescTexture (Mat input, int index, int level, int region, Mat
    desc){
480
481     double mat [SIZE][SIZE] = {0};
482     int start = 0, end = 3, index_desc = 0;
483     Mat mat_cv;
484
485     //imshow("input image", input);
486
487     copyToArr_2d(input, mat);
488
489     for (int i = 0; i < level; i++){
490
491         dyadic_2d (mat, SIZE);
492
493     }
494
495     //the calculation of dyadic using ordinary array in C,
496     //have to change to Mat of opencv
497     mat_cv = Mat (SIZE, SIZE, CV_64FC1, mat);
498
499     if (region == 0){
500         start = 0;
501         end = 0;
502     }
503
504     else if (region == 1){
505         start = 1;
506         end = 3;
507     }
508
509     else if (region == 2){
510         start = 0;
511         end = 3;
512     }
513
514
515     for (int i = start; i <= end; i++){
516
```

```
517         Mat output_dyad = Mat (SIZE_ORI, SIZE_ORI, CV_64FC1);
518
519         waveDetailDyad(mat_cv, output_dyad, i);
520
521         waveletDesc(output_dyad, index, index_desc, desc);
522
523         index_desc += NUM_DESC_TEX;
524
525         //for visualization purpose only
526         //if (i == 0)
527         //output_dyad.convertTo(output_dyad, CV_8UC1);
528
529         //imshow ("output_dyad", output_dyad);
530
531         //waitKey(0);
532
533         output_dyad.release();
534     }
535
536 }
537
538 //2D dyadic looping
539 void dyadic_2d(double mat[][SIZE], int quarter){
540
541     int row, step;
542
543     for(step = 0; step < 2; step++){
544         for(row = 0; row < quarter; row++){
545             dyadic_2 (mat, quarter, row);
546         }
547
548         transpose (mat, quarter);
549     }
550 }
551
552 //2D dyadic wavelet transform
553 void dyadic_2 (double mat[][SIZE], int n, int row){
554
555     double *tempbank = 0;
556     double h1, h2, h3, h4, g1, g2, g3, g4, g5, g6;
557     int i, j;
558
559     j = 1;
560
561     if (tempbank == 0)
562         tempbank = (double *)malloc(n * sizeof (double));
563
564     for (i = 0; i < n; i++)
565         *(tempbank+i) = 0;
566
567
568     h1 = 0.125*sqrt(2.0);
569     h2 = 0.375*sqrt(2.0);
570     h3 = 0.375*sqrt(2.0);
571     h4 = 0.125*sqrt(2.0);
572
```

```
573     g1 = -0.03125*sqrt(2.0);
574     g2 = -0.09375*sqrt(2.0);
575     g3 = -0.5625*sqrt(2.0);
576     g4 = 0.5635*sqrt(2.0);
577     g5 = 0.09375*sqrt(2.0);
578     g6 = 0.03125*sqrt(2.0);
579
580
581     for (i = 1; i < (n/2)-2; i++){
582
583         if (i == 1){
584             tempbank[j*2] = (g1 * mat[row][n-1]) + (g2 * mat[row][i-1]) +
    (g3 * mat[row][i])+ (g4 * mat[row][i+1])+ (g5 * mat[row][i+2])+ (g6 *
    mat[row][i+3]);
585             tempbank[j*2+1] = (h1 * mat[row][i-1]) + (h2 * mat[row][i]) +
    (h3 * mat[row][i+1])+ (h4 * mat[row][i+2]);
586         }
587
588         else if (i == ((n/2)-3)){
589             tempbank[j*2] = (g1 * mat[row][i-2]) + (g2 * mat[row][i-1]) +
    (g3 * mat[row][i])+ (g4 * mat[row][i+1])+ (g5 * mat[row][i+2])+ (g6 *
    mat[row][0]);
590             tempbank[j*2+1] = (h1 * mat[row][i-1]) + (h2 * mat[row][i]) +
    (h3 * mat[row][i+1])+ (h4 * mat[row][i+2]);
591         }
592
593         else{
594             tempbank[j*2] = (g1 * mat[row][i-2]) + (g2 * mat[row][i-1]) +
    (g3 * mat[row][i])+ (g4 * mat[row][i+1])+ (g5 * mat[row][i+2])+ (g6 *
    mat[row][i+3]);
595             tempbank[j*2+1] = (h1 * mat[row][i-1]) + (h2 * mat[row][i]) +
    (h3 * mat[row][i+1])+ (h4 * mat[row][i+2]);
596         }
597
598         j++;
599     }
600
601     //handling edge problem because of number of filters used
602     //front edge problem
603     tempbank[0] = (g1 * mat[row][(n/2)-2]) + (g2 * mat[row][(n/2)-1]) + (
    g3 * mat[row][0])+ (g4 * mat[row][1])+ (g5 * mat[row][2])+ (g6 * mat[
    row][3]);
604     tempbank[1] = (h1 * mat[row][(n/2)-1]) + (h2 * mat[row][0]) + (h3 *
    mat[row][1])+ (h4 * mat[row][2]);
605
606     //end edge problem
607     tempbank[n-2] = (g1 * mat[row][(n/2)-4]) + (g2 * mat[row][(n/2)-3]) +
    (g3 * mat[row][(n/2)-2])+ (g4 * mat[row][(n/2)-1])+ (g5 * mat[row][0])+
     (g6 * mat[row][1]);
608     tempbank[n-1] = (h1 * mat[row][(n/2)-3]) + (h2 * mat[row][(n/2)-2]) +
    (h3 * mat[row][(n/2)-1])+ (h4 * mat[row][0]);
609
610     for (i = 0; i < n; i++){
611         if (i%2==0){
612             mat[row][n/2 + i/2] = tempbank[i];
613         }
```

```
614         else{
615             mat[row][i/2] = tempbank[i];
616         }
617     }
618 
619     delete tempbank;
620 }
621 
622 //transpose image for wavelet purpose
623 void transpose (double matrix[][SIZE], int quarter){
624 
625     double temp = 0.0;
626 
627     for (int i = 0; i < quarter; i++){
628         for (int j = i+1; j < quarter; j++){
629 
630             temp = matrix[i][j];
631             matrix[i][j] = matrix[j][i];
632             matrix[j][i] = temp;
633         }
634     }
635 }
636 
637 //copy opencv mat data to ordinary C array
638 void copyToArr_2d (Mat input, double matrix[][SIZE]){
639 
640     for (int i = 0; i < input.rows; i++){
641         for (int j = 0; j < input.cols; j++){
642             matrix[i][j] = input.at<uchar>(i,j);
643 
644         }
645     }
646 }
647 
648 //choose wavelet decomposed image details
649 void waveDetailDyad (Mat mat, Mat& output, int detail){
650 
651     /*
652      detail = 0 -> low-pass
653      detail = 1 -> horizontal
654      detail = 2 -> vertical
655      detail = 3 -> diagonal
656      */
657 
658     if (detail == 0){
659 
660         mat(Rect(0, 0, SIZE_ORI, SIZE_ORI)).copyTo(output);
661 
662     }
663 
664     else if (detail == 1){
665 
666         mat(Rect(SIZE_ORI, 0, SIZE_ORI, SIZE_ORI)).copyTo(output);
667     }
668 
669     else if (detail == 2){
```

```
670
671         mat(Rect(0, SIZE_ORI, SIZE_ORI, SIZE_ORI)).copyTo(output);
672
673     }
674
675     else{
676
677         mat(Rect(SIZE_ORI, SIZE_ORI, SIZE_ORI, SIZE_ORI)).copyTo(output);
678     }
679
680 }
681
682 //calculation of texture feature
683 void waveletDesc (Mat input, int index, int index_desc, Mat& output){
684
685     /*
686      * output index 1 energy
687      * index 2 mean
688      * index 3 std_dev
689      * index 4 coef_var
690      */
691
692     int count = 0;
693
694     for (int i = 0; i < input.rows; i++)
695         for (int j = 0; j < input.cols; j++){
696             if (input.at<double>(i,j) > 0.0){
697                 output.at<double>(index-1,index_desc) += pow(input.at<
    double>(i,j), 2.0);
698                 output.at<double>(index-1,index_desc+1) += input.at<double
    >(i,j);
699                 count++;
700             }
701         }
702
703     output.at<double>(index-1,index_desc+1) /= count;
704
705     for (int i = 0; i < input.rows; i++)
706         for (int j = 0; j < input.cols; j++){
707             if (input.at<double>(i,j) > 0.0){
708                 output.at<double>(index-1,index_desc+2) +=
709                 pow((input.at<double>(i,j) -
710                       output.at<double>(index-1,index_desc+1)), 2.0);
711             }
712         }
713
714     output.at<double>(index-1,index_desc+2)
715     /= count;
716
717     output.at<double>(index-1,index_desc+2) =
718     sqrt(output.at<double>(index-1,index_desc+2));
719
720     output.at<double>(index-1,index_desc+3) =
721     (output.at<double>(index-1,index_desc+1) == 0.0) ? 0.0 :
722     ((output.at<double>(index-1,index_desc+2)/
723       output.at<double>(index-1,index_desc+1)) * 100.0);
```

```
724
725 }
726
727
728 //divide image become 4 regions and calculate its color histogram
729 void subImage(Mat src, int spoint, int epoint, int size, int index, Mat
        color_desc){
730
731     Mat subimage_1 = Mat (src.rows, src.cols, CV_8UC3);
732     Mat subimage_2 = Mat (src.rows, src.cols, CV_8UC3);
733     Mat subimage_3 = Mat (src.rows, src.cols, CV_8UC3);
734     Mat subimage_4 = Mat (src.rows, src.cols, CV_8UC3);
735
736     subimage_1 = src (Rect(spoint, epoint, size, size));
737
738     epoint += size;
739     subimage_2 = src (Rect(spoint, epoint, size, size));
740
741     spoint = size; epoint = 0;
742     subimage_3 = src (Rect(spoint, epoint, size, size));
743
744     spoint = size; epoint = size;
745     subimage_4 = src (Rect(spoint, epoint, size, size));
746
747     int feature_num = 0;
748
749     calc_histo(subimage_1, feature_num, index, color_desc);
750     feature_num += NUM_BINS;
751     calc_histo(subimage_2, feature_num, index, color_desc);
752     feature_num += NUM_BINS;
753     calc_histo(subimage_3, feature_num, index, color_desc);
754     feature_num += NUM_BINS;
755     calc_histo(subimage_4, feature_num, index, color_desc);
756
757 }
758
759 //HSV color histogram
760 void calc_histo(Mat subimage, int feature_num, int index, Mat color_desc){
761
762     /*display the window name with pattern: subimage + region
763      region:
764      |1 3|
765      |2 4|
766      */
767
768     Mat hsv(subimage);
769
770     cvtColor(hsv, hsv, CV_BGR2HSV);
771
772     //let's quantize the hue to 6 levels
773     //and the saturation to 4 levels
774     int hbins = 6, sbins = 4;
775     int histSize[] = {hbins, sbins};
776
777     //hue varies from 0 to 150, see cvtColor
778     //combination from red -> yellow -> green
```

```
779     float hranges[] = { 0, 150 };
780 
781     //saturation varies from 0 (black-gray-white) to
782     //255 (pure spectrum color)
783     float sranges[] = { 0, 255 };
784 
785     const float* ranges[] = { hranges, sranges };
786     MatND hist;
787 
788     //we compute the histogram from the 0-th and 1-st channels
789     int channels[] = {0, 1};
790 
791     calcHist( &hsv, 1, channels, Mat(),// do not use mask
792               hist, 2, histSize, ranges,
793               true, //the histogram is uniform
794               false );
795 
796     for( int h = 0; h < hbins; h++ )
797         for( int s = 0; s < sbins; s++ ){
798             color_desc.at<double>(index-1, feature_num) = hist.at<float>(h
    ,s);
799             feature_num++;
800         }
801 
802 }
803 
804 //append all features to one file
805 void appendAllDesc (Mat shp1, Mat shp2, Mat tex, Mat col, Mat output){
806 
807     shp1.copyTo(output(Rect(0, 0, shp1.cols, shp1.rows)));
808 
809     shp2.copyTo(output(Rect(shp1.cols,0,shp2.cols,shp2.rows)));
810 
811     tex.copyTo(output(Rect(shp1.cols+shp2.cols,0,tex.cols,tex.rows)));
812 
813     col.copyTo(output(Rect(shp1.cols+shp2.cols+tex.cols,0,col.cols,col.
    rows)));
814 
815 }
```